VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OSOBNÍ PLÁNOVAČ FINANCÍ PRO OS ANDROID

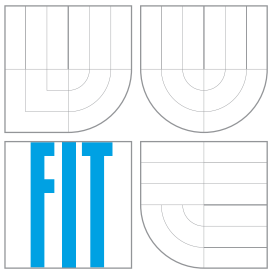
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN ODSTRČILÍK

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OSOBNÍ PLÁNOVAČ FINANCÍ PRO OS ANDROID

PERSONAL FINANCE MANAGER FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN ODSTRČILÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR POSPÍCHAL
SUPERVISOR

BRNO 2011

Abstrakt

Cílem bakalářské práce bylo vytvořit aplikaci k osobnímu plánování financí pro platformu Android. Taková aplikace by měla nabízet jejím uživatelům možnost zaznamenávat transakce, které reprezentují tok peněz, a tyto transakce uživateli prezentovat formou přehledů. Pomocí nich by pak uživatel měl nabýt kontroly nad jeho financemi. Práce je zaměřená také na tvorbu uživatelského rozhraní, který by mělo být intuitivní a působit moderně. Obsahem práce je také analýza existujících plánovačů osobních financí pro přední mobilní platformy iOS a Android, která je jako cílová platforma detailně popsána skrz její historii až po možnosti vývoje aplikací. Nedílnou součástí práce je také návrh, implementace a testování aplikace. Výsledkem celého snažení je funkční a nenáročný osobní plánovač financí pro mobilní zařízení poháněná Androidem.

Abstract

The goal of this bachelor thesis is to create an application for personal finance management running on Android platform. Such application should provide a possibility to record transactions representing personal money flow for its users, and to show reports from these transactions. The reports should provide capability to control user's finances. This work is also focused on implementation of application user interface, which should be intuitive and modern. Additionally, thesis contains analysis of existing personal finance managers available for major mobile platforms like iOS and Android. Android, as a targeted one, is detaily described through its history to developer matters. Other important parts of this work are design, implementation and debugging of final application. Described effort is resulting in a functional and lightweight personal finance manager for Android powered devices.

Klíčová slova

osobní plánovač financí, Android, uživatelské rozhraní, návrhové vzory, Java

Keywords

personal finance manager, Android, user interface, design patterns, Java

Citace

Martin Odstrčilík: Osobní plánovač financí pro OS Android, bakalářská práce, Brno, FIT VUT v Brně, 2011

Osobní plánovač financí pro OS Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Pospíchala. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Odstrčilík
16. května 2011

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce, panu Ing. Petru Pospíchalovi, za neocenitelnou pomoc a příkladné vedení při řešení bakalářské práce.

Obsah

1 Úvod **3**

2 Osobní plánování financí **5**
2.1 Využití osobního plánovače financí 5
2.2 Dostupné plánovače osobních financí 5
2.2.1 iXpenseIt 6
2.2.2 iBearMoney 6
2.2.3 Pageonce Pro – Money & Bills 6

3 Platforma Android **8**
3.1 Konkurence platformy Android 8
3.1.1 Platforma Symbian 8
3.1.2 Platforma BlackBerry 8
3.1.3 Platforma iOS 9
3.2 Počátek platformy Android 9
3.2.1 Historický přehled verzí Androidu 10
3.3 Vývoj aplikací pro Android 13
3.3.1 Architektura platformy 14
3.3.2 Virtuální stroj Dalvik 14
3.3.3 SDK a Eclipse 14
3.4 Stavební prvky aplikace 15
3.4.1 Activity 16
3.4.2 Services 16
3.4.3 Broadcast receivers 18
3.4.4 Komunikace komponent – Intents 18
3.4.5 Content Providers 18
3.4.6 Manifest 18
3.5 Odlišnosti verzí a hardwaru mobilních zařízení 18

4 Návrh aplikace **20**
4.1 Uživatelské rozhraní 20
4.2 Aktuální den 20
4.3 Přehledy transakcí 21
4.4 Uživatelské účty 22
4.5 Vytváření transakcí 22
4.6 Transakce 23
4.7 Kategorie 23
4.8 Uživatelský účet 23

4.9 Ukládání a práce s daty . . . 23
4.9.1 Diagram případů užití aplikace . . . 23
4.9.2 Diagram vztahů mezi entitami dat . . . 25

5 Implementace **26**
5.1 Struktura aplikace . . . 26
5.2 Jádro aplikace . . . 26
5.3 Aplikační data . . . 27
5.3.1 Třída `Database` . . . 27
5.3.2 Třída `Settings` . . . 28
5.3.3 Třída `Statistics` . . . 29
5.4 Implementace logických celků . . . 29
5.4.1 Přepínání logických celků . . . 30
5.4.2 Uživatelské rozhraní . . . 30
5.4.3 Vytvoření a editace transakcí . . . 31
5.4.4 Zobrazení transakce a přehledů . . . 32
5.4.5 Správce účtů . . . 34
5.4.6 Tabulka limitů . . . 34
5.5 Testování . . . 34

6 Závěr **36**
6.1 Přístup ke splnění zadání . . . 36
6.2 Výsledná aplikace . . . 36
6.3 Budoucnost a rozšíření aplikace . . . 37

Seznam příloh **42**

A Obsah DVD **43**

Kapitola 1

Úvod

Nechvalným faktorem poslední doby je rostoucí náročnost a uspěchanost každodenního života. Lidé mají čím dál tím větší problém s dodržováním naplánovaných událostí, často zapomínají a celkově nestíhají. Díky velmi rychlému koloběhu událostí každého dne je až nereálné si zapamatovat informaci o tom, kde, kdy a kolik jsme zaplatili za nákup v obchodě, kolik stála večeře v restauraci či kolik jsme utratili na dovolené. Dostáváme se tak do situace, kdy nemáme představu o stavu našich financí. Tato situace tak může být a často také bývá velmi frustrující. V některých případech může být až kritická, neboť zjistit například na vlakovém nádraží v době nutného odjezdu, že v peněžence je prázdno a na kartě není ani koruna, je velmi nepříjemné a problematické.

Z tohoto pohledu je dobré a výhodné mít možnost si informaci o stavu našich financí uchovávat. Můžeme tak činit pomocí různých technik či nástrojů. Někteří lidé si například skladují veškeré stvrzenky a platební účty, jiní si zase jednotlivé platby zapisují do deníků nebo využívají speciálních nástrojů a to jak ve formě různých zařízení, tak softwarových aplikací. Tuto možnost můžeme nazvat osobním plánováním financí.

V posledních letech naše společnost zažívá také rozmach na poli mobilních zařízení a to zejména chytrých mobilních telefonů a miniaturních přenosných osobních počítačů. Jejich cena již delší dobu není tak přehnaná, jak tomu bylo dříve, a tak se tato zařízení dostávají více a více z výloh obchodů do rukou koncových uživatelů. S rozvojem samotných mobilních zařízení dochází ke vzniku nových platforem a operačních systémů, které tyto telefony řídí. Jedním z průkopníků v tomto směru byl operační systém Symbian, dnes známý převážně z telefonů značky Nokia. Jednalo se z počátku o platformu využívanou především zkušenějšími a majetnějšími uživateli. Revoluci na poli mobilních platforem přinesl mobilní telefon iPhone společnosti Apple. Díky jeho průkopnickému ovládání a uživatelské přívětivosti se chytré mobilní telefony poprvé masově rozšířily mezi širokou veřejnost takřka po celém světě. Dramaticky rostoucímu vlivu iPhonů ale přišla do cesty platforma Android. Jedná se o nejrychleji se rozvíjející platformu posledních let, která nejen že náskok iPhonu a jeho operačního systému iOS dohnala, ale dokonce, co se zastoupení na trhu týče, i předběhla [8, 42]. Jako další zástupce mobilních platforem bych zmínil také Palm (nyní webOS) či Windows Phone 7 (dříve Windows Mobile). V současné době se ale nejedná o výraznější zástupce, byť svoje zastánce stále mají. K doplnění souvislostí lze doporučit přehledový článek [43], který se snaží vystihnout přednosti a nevýhody aktuálních mobilních platforem.

Působnost moderních mobilních platforem není zaměřena pouze na chytré mobilní telefony, ale jsou jimi cílená i jiná mobilní zařízení, jako jsou například dnes velmi oblíbené netbooky a tablety. Vznikají tak zařízení, která jsou schopná telefonovat, číst a posílat elektronickou poštu a zprávy, prohlížet internetový obsah, pořizovat fotografie a natáčet vi-

dea. Mnohé přístroje také navíc umožňují zprostředkovávat video hovory, číst elektronické knihy, zobrazovat mapy a navigovat v nich. Nechybí ani možnost rozšíření pomocí jiných zařízení a aplikací třetích stran. Zmíněné funkce jsou dnes převážně těmi základními a povinnými. Jsou však nabízeny ve stále lepší kvalitě, větším formátu a s delší výdrži. Tímto rapidním a všestranným rozmachem nových zařízení a platforem se otevírají další oblasti jejich využití a to nejen pro koncové uživatele, ale také vývojáře hardwaru a softwaru. Ti díky všestranným možnostem těchto zařízení mohou šířit své nápady a ideje dalším směrem a oslovit tak široké portfolio uživatelů a potenciálních platících zákazníků.

Logickým vyústěním zmíněných přístupů se pak jeví využití mobilního zařízení pro plánování financí. Právě návrhem a implementací takové aplikace se zabývá tato práce. V této technické zprávě budou postupně popsány možnosti plánování osobních financí a jeho využití. Kapitola 2 je doplněna ukázkami předně používaných aplikací. V úvodu kapitoly 3 jsou stručně popsány přední platformy pro mobilní zařízení. Jádro této kapitoly je pak tvořeno detailním popisem cílené platformy Android zahrnující náhled na možnosti a vhodné postupy vývoje aplikací. Zbývající část technické zprávy je věnována návrhu (kapitola 4 a implementaci (kapitola 5) vytvářené aplikace. Samotný konec zprávy se v kapitole 6 snaží shrnout celkový průběh a výsledek práce.

V textu kapitol je z důvodu přehlednosti použito zvýraznění následujícím způsobem. *Kurzívou* jsou psány názvy komponent Androidu, logických celků aplikace a anglických termínů a názvů technologií. `Strojovým písmem` jsou značeny názvy objektů, tříd a jejich metod.

Kapitola 2

Osobní plánování financí

Následující kapitola se zabývá možnostmi využití osobního plánování financí v každodenním životě člověka. Dále komentuje předpokládanou a vhodnou funkcionalitu plánovače určeného k tomuto účelu. Následně jsou shrnuty vlastnostmi předních používaných aplikací.

2.1 Využití osobního plánovače financí

Osobní plánování financí nám pomáhá spravovat naše příjmy a výdaje. K úspěšné správě je potřeba veškeré transakce zaznamenávat a z těchto záznamu vytvářet přehledy. Pro efektivní správu je potřebné mít více pohledů na sesbírané údaje. Proto se nelze omezit pouze na jeden formát prezentace. Součástí variability zobrazení údajů musí být bezpochyby i výběr zobrazovaných informací. Ne vždy nás zajímají kompletní výpisy, které by navíc působily nepřehledně. Mezi hlavní funkce aplikace bezpochyby patří přidávání a zobrazování transakcí. Další vhodnou funkcí je možnost nastavení omezení v podobě limitů hodnot transakcí za různá období (dny, týdny či měsíce). Příjemnou vlastností plánovače je možnost vytváření více uživatelských účtů. Ty nabývají užitečnosti v případech, kdy potřebujeme označit spolu související plateby (osobní výdaje, firemní příjmy či výdaje v zahraničí). S výdaji v zahraničí souvisí podpora jiných měn a jejich převodů. Důležitou funkcí je také možnost určité transakce naplánovat a případně nastavit upozornění na jejich výskyt. Neméně významnou, ne-li klíčovou vlastností aplikace, je také její uživatelská přívětivost. Rychlost přidání nové transakce by měla být otázkou několika málo vteřin a obsluha by měla být jednoduchá a intuitivní tak, aby samotná práce s aplikací uživatele neodradila od jejího používání.

2.2 Dostupné plánovače osobních financí

V současné době se na předních příčkách v oblíbenosti a prodeji v kategorii finančních aplikací objevují vedle aplikací pro přehled finančního trhu právě aplikace pro správu osobních financí. Tyto aplikace převážně nabízejí možnost přidávání transakcí se specifikací času a místa vzniku transakce. U transakce je většinou také možné zvolit způsob úhrady (hotově, kreditní kartou či převodem) a účel vzniku (nákup potravin, občerstvení, platba za energie, dary, atd.). Aplikace podporují zobrazení a filtraci přehledů transakcí za určitá období a to dle způsobu nebo účelu transakce. Součástí aplikací bývá možnost vytváření grafických přehledů či zpráv a jejich následný tisk nebo zaslání elektronickou poštou. Obvyklá u těchto aplikací je i podpora více uživatelských účtu. Některé aplikace jsou synchronizo-

vány s online službou, která poskytuje uživateli jak možnost zobrazení údajů z aplikace přes internetový prohlížeč, tak možnost zálohování. Aplikace nabízejí také většinou možnost provázání s bankovním účtem či kreditní kartou. Při přidávání transakce je pak možné vybrat účet či kreditní kartu, ke které se transakce vztahuje a tím si hlídat pohyb peněz na těchto finančních úložištích.

2.2.1 iXpenseIt

iXpenseIt je uživateli a světovými magazíny dobře hodnocená aplikace pro platformu iOS. Mezi význačné funkce této aplikace patří tisk přehledů a zpráv za jednotlivá období, podpora více měn společně s převodníkem, kalkulačka, plánování rozpočtu, grafické zobrazení příjmů, výdajů a přehledů, možnost nastavení skupin a podskupin transakcí. Aplikace také podporuje zálohování a možnost zabezpečení heslem [46]. Ukázka vzhledu aplikace je k vidění na obrázku 2.1.

Obrázek 2.1: Aplikace iXpenseIt pro platformu iOS [45]

2.2.2 iBearMoney

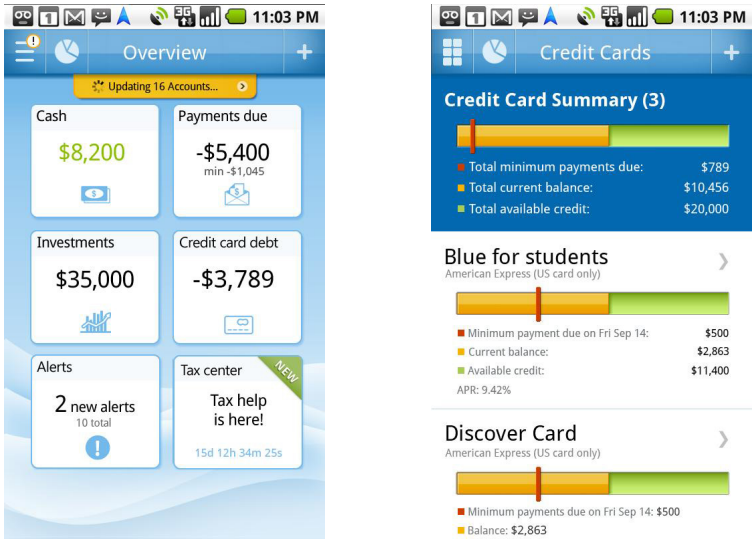
Tato aplikace se mimo svůj vzhled pyšní také umístěním na prvních pěti příčkách v prodeji ve 30 státech (údaj z února 2011) světa [40]. Přednostmi aplikace jsou velmi rozsáhlé a pokročilé možnosti plánování rozpočtu, neomezený počet uživatelských účtů (peněženek), rychlý způsob přidávání nových transakcí, podpora více měn s aktualizací kurzů, detailní grafické přehledy pro výdaje, příjmy a celkový tok peněz. Výhodou iBearMoney je schopnost synchronizace dat mezi více instancemi aplikace na různých iOS zařízeních. Aplikace je dostupná jak pro platformu iOS, tak i pro platformy Android a Windows Mobile [40]. Vzhled aplikace v mutaci pro tablet iPad společnosti Apple je k nahlédnutí na obrázku 2.2.

2.2.3 Pageonce Pro – Money & Bills

Jedná se o nejoblíbenější placenou aplikaci pro osobní plánování financí pro platformu Android [48]. Tato aplikace se vyznačuje uživatelsky přívětivou úvodní obrazovkou, která

Obrázek 2.2: Aplikace iBearMoney v mutaci pro Apple iPad [39]

zobrazuje nejdůležitější finanční informace a ovládací prvky. Aplikace dále nabízí možnost upozornění na platby, souhrnné a přehledové grafické zprávy, asociaci s bankovními účty, kartami a sledování průběhů investic. Součástí tohoto řešení je i webový portál, se kterým jsou data aplikace synchronizována. Novou funkcí v poslední verzi aplikace je přepočet informací z čítačů telefonu (počet odeslaných SMS zpráv, provolaných minut a přenesených dat) na výdaje. Aplikace Pageonce – Money & Bills je dostupná pro většinu majoritních mobilních platforem (Android, iOS, BlackBerry a Windows Phone) [47]. Obrázek 2.3 ukazuje intuitivní uživatelské rozhraní aplikace.



Obrázek 2.3: Mutace aplikace Pageonce Pro – Money & Bill pro platformu Android [47]

Kapitola 3

Platforma Android

V této kapitole je blíže představena platforma Android. V úvodní části jsou přehledově představeni její konkurenti. Dále je objasněn vznik a historie. Ve zbývající části je pozornost věnována způsobům a metodice vytváření aplikace. Jsou také popsány základní komponenty, možnosti a důležité aspekty, se kterými by měl vývojář při tvorbě aplikace počítat.

3.1 Konkurence platformy Android

Již v úvodní kapitole bylo nastíněno, že mobilních platforem existuje několik. Mezi jednotlivými zástupci lze nalézt dobře známé platformy Symbian a BlackBerry, které ale v poslední době začínají ztrácet na oblibě a celkovém využití. Do popředí se staví relativně mladí zástupci, mezi které nepochybně patří platforma iOS společnosti Apple a Android společnosti Google. Vývoj postavení zmíněných předních mobilních platforem je ilustrován grafem 3.1. Ten zobrazuje celosvětové statistiky rozložení přístupů z mobilních operačních systémů ke sledovaným stránkám a aplikacím za poslední 3 roky. Údaje statistik jsou pořizovány analytickým systémem na bázi prokliků StatCounter [53].

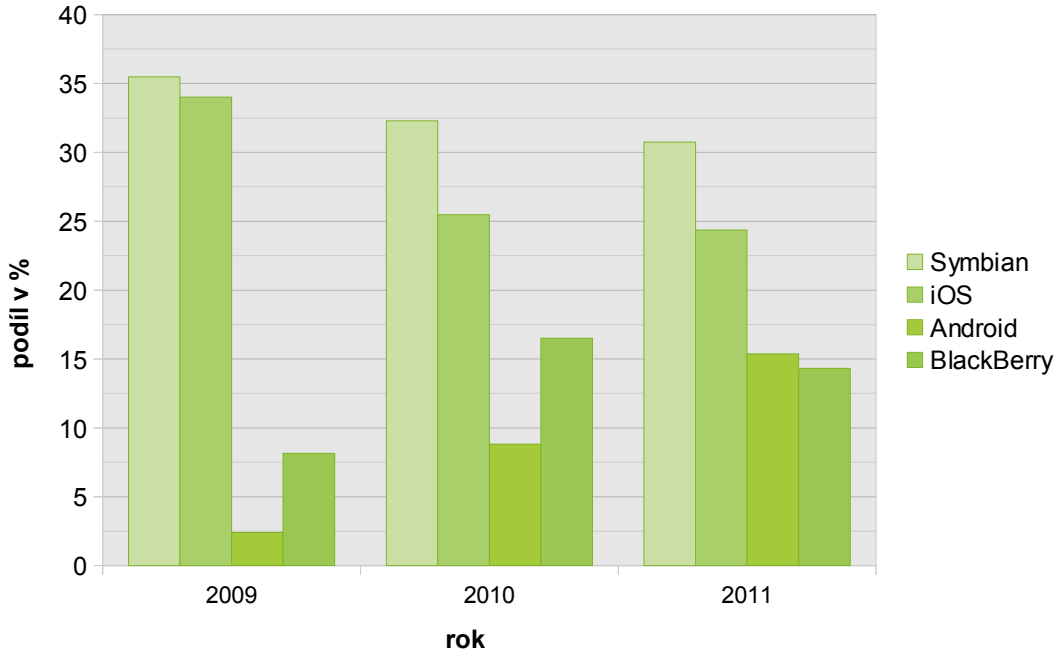
3.1.1 Platforma Symbian

Symbian si jako jeden z nejstarších zástupců operačních systémů pro chytré mobilní telefony stále drží úctyhodnou pozici nejvyužívanějšího. Do jisté míry je tomu tak díky společnosti Nokia, která se pyšní světovým prvenstvím v prodeji mobilních telefonů [41] a nyní stojí za vývojem této platformy [54]. Symbian je stálicí mobilních platforem, která ale každým dnem ztrácí na své oblíbenosti [53]. Aktuální verzí je Symbian∧3. Z programátorského hlediska je možné vyvíjet aplikace pro tuto platformu v nativním jazyce Symbian C++, Qt, Pythonu či Java ME [50].

3.1.2 Platforma BlackBerry

Mobilní zařízení BlackBerry kanadské firmy RIM se těší oblibě hlavně v Severní Americe [53]. Přednostmi platformy BlackBerry jsou především dobrá integrace s různými komunikačními prostředky (elektronická pošta, IM komunikace a sociální sítě). Zařízení BlackBerry jsou převážně vyhledávány společnostmi a náročnějšími uživateli kvůli rozsáhlým možnostem spolupráce (plánování událostí, vytvářením úkolů a poznámek) a komunikace s kolegy či přáteli. Smyslem tohoto propojení je možnost globální a bezpečné synchronizace mezi účastníky. Typickým rysem telefonů BlackBerry je plnohodnotná QWERTY klávesnice,

jež uživatelům usnadňuje textovou komunikaci. Vývojáři aplikací mají na výběr několik přístupů, mezi které patří například vývoj v jazyce Java ME nebo pomocí BlackBerry Web Development [3].



Obrázek 3.1: Statistika přístupů předních mobilních operačních systémů na monitorované stránky a aplikace v posledních třech letech [53]

3.1.3 Platforma iOS

V případě platformy iOS společnosti Apple (dříve iPhone OS) lze hovořit o revoluci na poli mobilních platforem. Vizí této platformy je ovládání zařízení kompletně pomocí dotyků a gest. Zařízení se vyznačují velmi malým počtem hardwarových tlačítek a spoustou senzorů. Unikátním prvkem ovládání je možnost provádět příkazy také polohováním zařízení v prostoru a to díky akcelerometru a gyroskopy zaznamenávajícími tyto změny. Typickým příkladem využití je možnost vyvolání akce aplikace pouhým zatřepáním se zařízením. Mezi hlavní zástupce této platformy patří mobilní telefon iPhone, multimediální přehrávač iPod Touch a tablet iPad. Vývojovým jazykem je zde Objective-C. Aktuální verzí operačního systému je iOS 4.3. Apple je znám svou striktní politikou, co se týče aplikací třetích stran. Z tohoto důvodu je oficiálně možné aplikace distribuovat pouze přes Apple App Store a může se stát, že vytvořená aplikace bude zamítnuta [50, 1].

3.2 Počátek platformy Android

Impulsem ke vzniku platformy Android byla skutečnost, že do té doby existující mobilní platformy byly příliš omezené jen na úzké spektrum působnosti a celkově zaostávaly za vývojovými platformami pro stolní počítače. Tuto skutečnost se Google snažil změnit, a tak za úplný počátek platformy Android lze považovat rok 2005, kdy společnost Google koupila začínající firmu Android, která do té doby vyvíjela tento operační systém.

Dalšími krokem k založení dnešní podoby platformy Android bylo vytvoření sdružení Open Handset Alliance (dále OHA), které je v čele se společností Google tvořeno dalšími předními a světovými výrobci softwaru, hardwaru a mobilními operátory. Sdružení se začalo formovat v roce 2007. Mezi prvními se připojili společnosti zvučných jmen, jakými jsou například T – Mobile, Samsung, Sony Ericsson, Intel a Texas Instruments. V současné době je společenství tvořeno již 80 společnostmi [20]. Tento počet jasně hovoří o tom, že Android má silné zázemí a podporu. Sdružení OHA se stará jak o rapidní vývoj samotného operačního systému Android, tak o rozvoj a inovaci mobilních zařízení, mezi které se neřadí pouze mobilní telefony, ale také například netbooky a tablety. Snahou OHA je poskytnout uživatelům lepší dojem, více užitku a také zábavy plynoucí z používání těchto zařízení. OHA však myslí i na potřeby operátorů, výrobců mobilních zařízení a vývojářů. Prostředkem k dosažení těchto cílům má být otevřená mobilní platforma – Android [21, 52].

Již zmíněné a další klíčové momenty vzniku a rozvoje platformy lze znázornit přehledovým obrázkem 3.2.

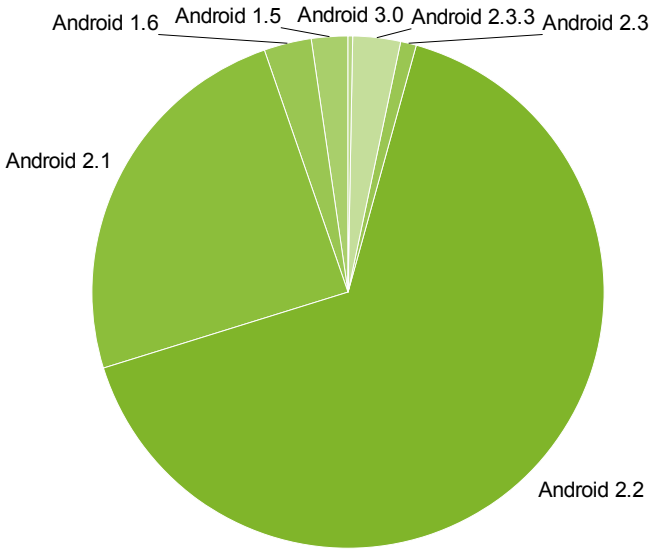
Obrázek 3.2: Klíčové momenty počátku a následného vývoje platformy Android [52]

3.2.1 Historický přehled verzí Androidu

Rapidní vývoj platformy Android je možné demonstrovat na přehledu jednotlivých verzí, které rozšiřují základní verzi Androidu 1.0. Ta byla společně s vývojovým prostředím představena v září 2008 (viz obr. 3.2). Z pohledu nasazení jednotlivých verzí ve fyzických zařízeních jsou podstatné až verze Androidu 1.5 a vyšší. Předchozí verze jsou označeny jako zastaralé a nemají již praktické využití. V současné době verze 2.1 a vyšší zaujímají již 90% aktivních zařízení s platformou Android. Tento fakt podtrhuje převzatý graf na obrázku 3.3. Údaje byly pořízeny z mobilních zařízení, které se připojily k službě Android Market v období 18. dubna až 2. května 2011 [35].

Verze 1.5 - Cupcake

Verze Cupcake byla představena v dubnu 2009. Mezi její hlavní přínosy řádíme možnost nahrávání a přehrávání videa pořízeného pomocí integrované kamery a softwarovou klávesnici. Další zdokonalení se týkala především systémových aplikacím a uživatelského rozhraní (přidány animace přechodů). Byly přidány rozšíření umístitelné na úvodní obrazovku zařízení. Vylepšení doznala také komunikace pomocí technologie Bluetooth a GPS lokalizace [13].



Obrázek 3.3: Rozložení verzí Androidu na trhu – Android 1.5 – 2,3 %, Android 1.6 – 3,0 %, Android 2.1 – 24,5 %, Android 2.2 – 65,9 %, Android 2.3 – 1 %, Android 2.3.3 – 3 %, Android 3.0 – 0,3 % [35]

Verze 1.6 - Donut

Novinkou této verze byla integrace panelu pro rychlé vyhledávání a to jak v systémových souborech, tak informací na internetu. Vylepšení se týkala také kamery a galerie, kterou přinesla předchozí verze. Přibyla možnost využití VPN (*Virtual Private Network*) a indikátor stavu baterie. Výraznější změny se také dotkly aplikace Android Market především v podobě přepracování a rozdělení uživatelského rozhraní a jeho částí do příjemnější a efektivnější podoby. Verze 1.6 byla uvedena v září 2009 [14].

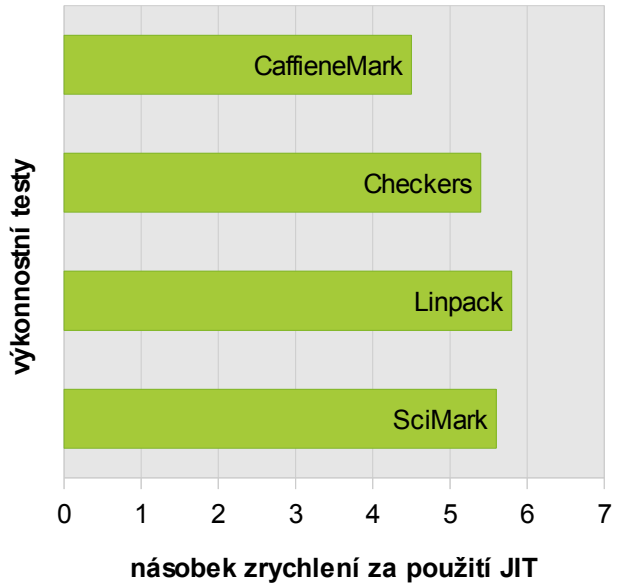
Verze 2.0/2.1 - Eclair

Verze 2.0, kterou velmi brzy vystřídala verze 2.1, přinesla z velké části pouze úpravy a zdokonalení stávajících částí systému. Byla přidána doposud chybějící možnost vytvoření více účtů pro elektronickou komunikaci společně se synchronizací nastavených účtů a to včetně Microsoft Exchange. Pozornost byla věnována i propojení kontaktů s jinými aplikacemi. Podstatnou změnu zaznamenal i internetový prohlížeč, který s novou verzí začal podporovat jazyk HTML ve verzi 5. Ta přináší podstatné zlepšení při integraci videa do internetových stránek a vyplňování formulářů. Jednotlivé verze byly uvedeny v říjnu 2009 (Android 2.0) a lednu 2010 (Android 2.1) [15].

Verze 2.2 - Froyo

Velmi očekávanou změnou, jež tato verze přinesla, byla možnost instalace nových aplikací do externího úložiště – paměťové karty. Doposud se veškeré aplikace instalovaly do vnitřní paměti telefonu a uživatelé tak byli značně omezení její kapacitou. Froyo následuje vylepšení dostupná z předchozí verze a přidává možnosti zabezpečení telefonu heslem nebo vzdáleným zablokováním a smazáním obsahu zařízení. Většímu využití se dostává i Wi-Fi adaptérům. Ty lze od této verze využít jako vysílače a telefon tak může sloužit jako přístupový

bod pro jiná zařízení. Dalšímu vylepšení doznalo uživatelské rozhraní, konkrétně správci kontaktů, který byl kompletně přepracován, či přepínání vícejazyčných rozložení softwarové klávesnice. Přibyla podpora více rozlišení a velikostí displejů. V této verzi došlo i ke změně kompilátoru virtuálního stroje Dalvik 3.3.2 a díky tomu k nárůstu výkonu a zlepšení odezvy systému oproti předchozí verzi. Následující graf 3.4 zobrazuje násobek nárůstu výkonu oproti předchozí verzi 2.1 Eclair. K testování výkonu byly použity speciální testovací programy, které například měří výkonnost zpracování operací v plovoucí desetinné čárce (FLOPs) a práce s maticemi. Verze Froyo byla představena v květnu 2010 na konferenci Google I/O [16].



Obrázek 3.4: Nárůst výkonu za použití JIT oproti verzi 2.1 Eclair [10]

Verze 2.3 - Gingerbread

Nejaktuálnější verzí Androidu pro klasické mobilní telefony je verze 2.3, která se snaží držet krok s trhem těchto zařízení a jejich standardy. Rozšiřuje proto schopnosti systému o obsluhu nových senzorů či více kamer, kterými jsou nejnovější telefony vybaveny. Přibývá podpora SIP (*Session Initiation Protocol*) protokolu a technologie pro komunikaci s jinými elektronickými zařízeními na krátkou vzdálenost NFC (*Near Field Communication*). Dochází dále také k celkové optimalizaci systému, zjednodušení uživatelského rozhraní a opravě chyb předchozích verzí. Prvním telefonem s touto verzí bylo zařízení přímo od společnosti Google - Nexus S. Jedná se o nástupce předchozího modelu Nexus One. Tyto telefony se vyznačují tím, že jejich uživatelské rozhraní není nijak upravováno výrobcem samotného zařízení, jak je zvykem u ostatních výrobců jako jsou HTC (Sense), Samsung (TouchWiz), LG či Motorola. Verze Gingerbread byla uvedena v prosinci 2010. Poslední verzí řady Gingerbread je 2.3.4, jež přidává do telefonů Nexus S a aplikace Gtalk videochat [36, 19, 38].

Verze 3.0 - Honeycomb

Verze Honeycomb je první verzí Androidu, která reaguje na rozšiřující se počet tabletů a jim podobných zařízení. Honeycomb je tak primárně určen pro tato zařízení, avšak použití na klasickém mobilním telefonu se nevylučuje. Tablety obecně dosahují větších rozměrů a mají tedy i větší displej, který je tak schopen na jedné obrazovce pojmout více informací. Z tohoto důvodu bylo ve verzi 3.0 kompletně přepracováno uživatelské rozhraní pro podporu těchto zařízení. Honeycomb přináší také podporu externích periférií připojitelných přes USB rozhraní, vícejádrových procesorů a výkonnějších grafických čipů. S nárůstem výkonnostního potenciálu těchto zařízení nabízí Honeycomb nově i hardwarovou akceleraci grafického standardu OpenGL. Ačkoli by se mohlo zdát, že verze 3.0 je úplně něco nového a jiného, opak je pravdou. Zařízení poháněná verzí Honeycomb jsou stále schopna spouštět aplikace ze starších verzí Androidu. Je tedy zachována zpětná kompatibilita. Vývojářům se však otevírají nové možnosti [17]. Obrázek 3.5 zobrazuje možný vzhled domovské stránky tabletu s verzí Android Honeycomb.

Obrázek 3.5: Domovská stránka zařízení s verzí Androidu Honeycomb [11]

3.3 Vývoj aplikací pro Android

Aplikace pro platformu Android jsou psané v jazyce Java. Nejedná se o klasickou Javu vycházející ze standardů Java SE ani verze pro mobilní telefony Java ME. Není tedy s těmito standardy plně kompatibilní a není tedy schopna využívat všech knihoven těchto standardů (obzvláště AWT a SWING). Rozdíl oproti klasické Javě není pouze v jazyce samotném, ale i v překladači a výsledném spustitelném kódu, viz kap. 3.3.2. Jednotlivé verze Androidu s sebou nesou i verzi programátorského rozhraní (API), pomocí kterého je aplikace vyvíjena.

Každá verze API definuje množinu knihoven a jejich metod, které jsou vývojářům dostupné. Knihovny jednotlivých verzí API společně s nástroji pro překlad, ladění a testování tvoří vývojový balíček (SDK), který je v současné době oficiálně plně podporován ve vývojovém prostředí Eclipse 3.3.3. Součástí SDK je i rozsáhlá dokumentace, která popisuje jednotlivé knihovny, jejich třídy, metody a způsob jejich použití. Tato dokumentace je doplněna množstvím praktických příkladů a ukázek zdrojových kódů. Vývoj aplikací pro platformu Android je možný na všech aktuálních operačních systémech - Windows, Linux a Mac OS X. Součástí podpory pro vývojáře není pouze dokumentace, ale také rozsáhlá komunita, se kterou může vývojář komunikovat pomocí diskuzních fór či skupin. Platforma Android a veškeré její součásti jsou volně dostupné a šiřitelné pod licencí Apache 2.0 a GPLv2.

3.3.1 Architektura platformy

Architekturu platformy Android je možné charakterizovat jako balík spolupracujících programů. Řadí se mezi ně především operační systém a jeho jádro, systémové a aplikační knihovny, prostředí pro běh aplikace a předpřipravené aplikace určené k zajištění klíčové funkcionality zařízení [18, 52]. Schéma architektury je ilustrováno obrázkem 3.6.

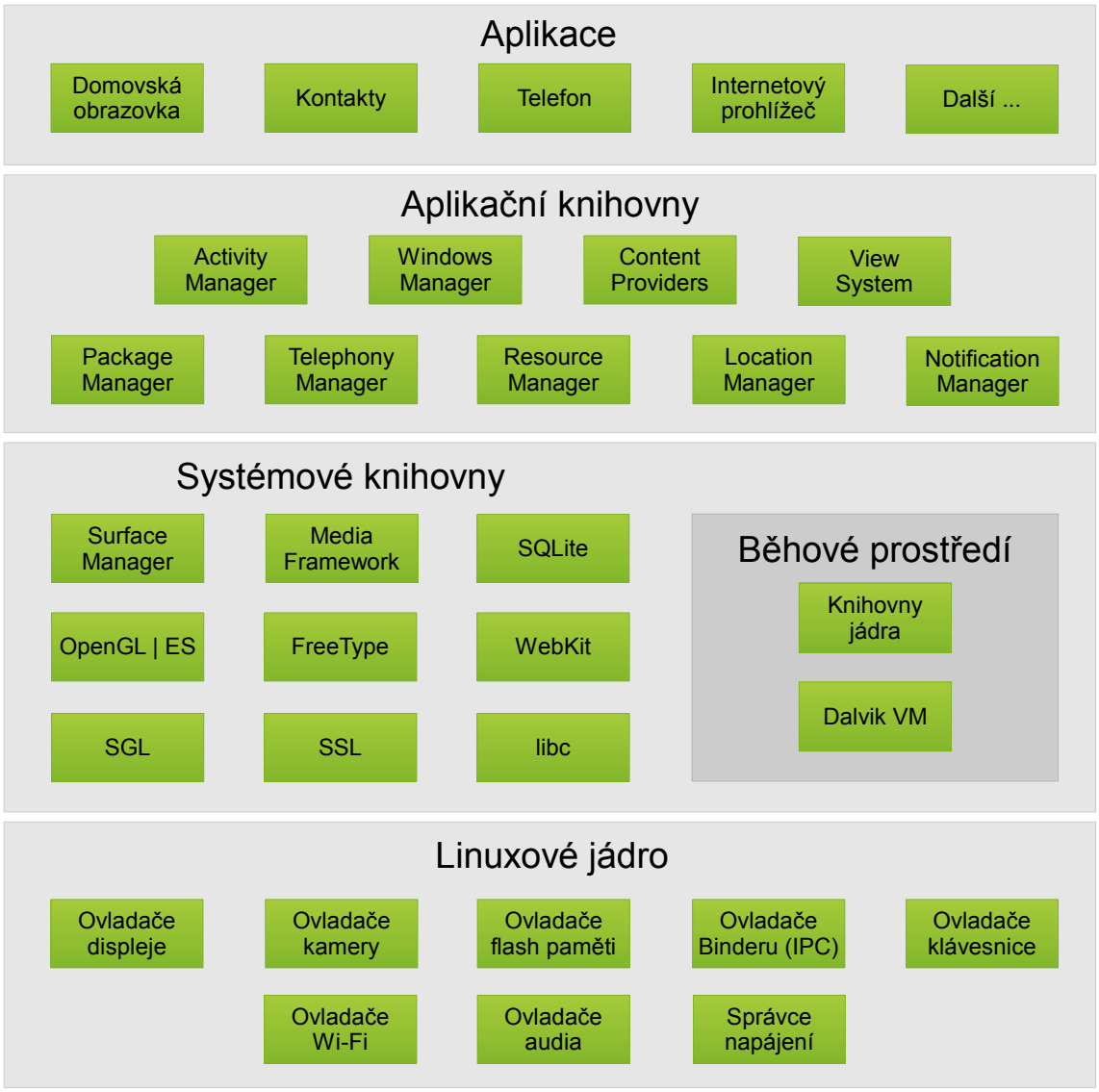
3.3.2 Virtuální stroj Dalvik

Spuštěná aplikace běží odděleně od ostatních ve vlastním virtuálním stroji. Přestože jsou aplikace pro Android napsány v jazyce Java, není pro jejich běh použit klasický virtuální stroj Javy (JVM), ale upravený a odlehčený virtuální stroj Dalvik. Jeho implementace vychází z podmnožiny knihoven otevřené implementace Javy – Apache Harmony [7]. Hlavní odlišností od JVM je v přístupu k vykonávání zdrojového kódu a zpracovávaným datům. JVM používá zásobník, kdežto Dalvik je čistě registrový stroj. K vykonávání kódu nepotřebuje rozsáhlou množinu instrukcí, jak je tomu u strojů využívajících zásobník. Jeho běh pak v paměti nezabírá tolik cenného místa. Dále se ve virtuálním stroji Dalvik nepoužívá klasický bytový kód Javy, ale je použita vlastní implementace [52, 18]. Od verze Androidu 2.2 využívá Dalvik JIT (just-in-time) kompilaci. Při běhu aplikace dochází tímto přístupem k interpretaci dynamických částí programového kódu. Statické části kódu jsou přeloženy před spuštěním aplikace a uloženy do vyrovnávací paměti. Tento hybridní přístup podstatně zrychluje běh aplikací a samotného systému, jak je znázorněno v grafu 3.4.

3.3.3 SDK a Eclipse

Vývoj aplikace pro platformu Android je doporučováno provádět v přenositelném vývojovém prostředí Eclipse, do kterého lze pomocí přidavného modulu integrovat SDK (viz kap. 3.3). Pomocí něj lze zvolit, které verze API (viz kap. 3.3) bude vývojář používat. Součástí SDK jsou šablony pro vytvoření nového projektu, které vývojáři pomáhají nadefinovat nastavení a pravidla pro vývoj, překlad a testování vytvářené aplikace. SDK také obsahuje nástroje pro grafické návrhy uživatelských rozhraní či správu grafických a datových konstant použitelných v samotné aplikaci. Vývojář může zvolit čistě programátorský přístup vytváření aplikace anebo využít právě zmíněné pomocné nástroje. Spuštěnou aplikaci je možné sledovat pomocí ladícího nástroje. Vývojář má v jeho prostředí přístup k obsahu paměti, aktuálně prováděné instrukci, ladícím výpisům, obsahům registrů a proměnných, spuštěným vláknům a také souborům aplikace. Testování aplikací je možné provádět na fyzických zařízeních připojených k počítači přes USB rozhraní nebo ve virtuálních zařízeních, která lze nastavit tak, aby přesně simulovala vlastnosti reálného zařízení. Nástroje SDK

přístupné z vývojového prostředí Eclipse jsou dostupné také přes systémovou příkazovou řádku či terminál [29].



Obrázek 3.6: Schéma architektury platformy Android [18]

3.4 Stavební prvky aplikace

Aplikace pro Android se skládá z několika základních komponent. Samotná aplikace neobsahuje jediný vstupní bod (např. funkci `main()`), který by byl při spuštění aplikace vykonán, ale dochází ke spuštění výchozí nebo posledně spuštěné části aplikace. Aplikace je tedy schopna uchovávat svůj stav a také z něj pokračovat. Každá aplikace má přidělený unikátní identifikátor, pomocí kterého k ní a jejím komponentám mohou ostatní aplikace přistupovat. Aplikace má také nadefinována přístupová práva vymezující možnosti přístupu k ní. Přistupující aplikace pak mají buď přístup k celé aplikaci nebo pouze jejím částem. V ná-

sledujících podkapitolách jsou jednotlivé části (komponenty) přiblíženy.

3.4.1 Activity

Z pohledu uživatele se tato komponenta jeví jako nejdůležitější, neboť právě s ní uživatel v aplikaci pracuje. *Activity* má tedy na starost interakci s uživatelem. Prezentuje uživateli data, přijímá od uživatele vstupy a tyto vstupy následně na základě účelu zpracovává. Aplikace může být celá tvořena buď jednou *Activity*, anebo může být rozdělena do více částí, které si v tomto případě předávají řízení. Uživatel však v jediném okamžiku pracuje právě s jednou [22].

Vzhled Activity – View

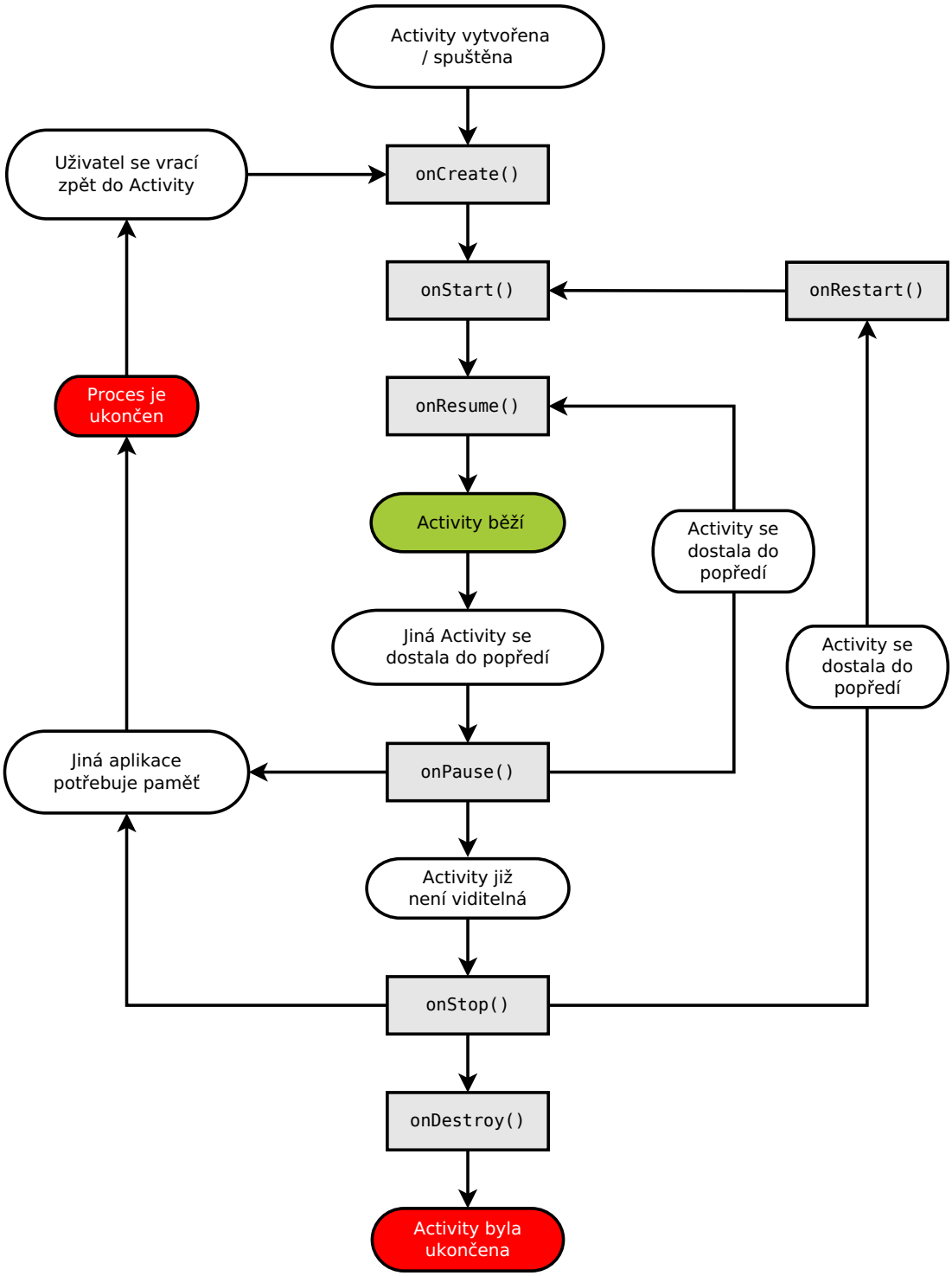
Samotná *Activity* poskytuje uživateli pouze data a prostředky, jak s nimi manipulovat. Vzhled jejich prezentace a tedy uživatelského rozhraní je definován až vybráním třídy vzhledu `View`, kterým se má *Activity* řídit. Třída `View`, případně její rozšíření, definují, zda budou data zobrazena v tabulkách, záložkách, mřížce, seznamech nebo se budou obecně skládat za sebe námi definovanými pravidly. Dále můžeme vytvářet tlačítka, vstupní textová pole či zatrhávací a přepínací tlačítka. SDK 3.3.3 poskytuje vývojáři několik předdefinovaných zobrazení a komponent pro zpracování uživatelských vstupů. Tyto prvky obsahují atributy, kterými lze specifikovat jejich vzhled, obsah a vlastnosti. Vývojář tak může použít implicitní definice rozložení, komponent a grafických prvků anebo si vytvořit vlastní, ať rozšířením již existujících, anebo vytvořením zcela nových [34].

Životní cyklus Aktivity

Activity se během svého života nachází v několika možných stavech. Změna stavu může nastat na základě akce uživatele nebo samotného systému. Typickými případy jsou přepnutí kontextu mezi *Activity* či aplikacemi, pozastavení a ukončení celé aplikace. Vývojář aplikace by měl jednotlivé úkony rozdělit mezi tyto stavy tak, aby při přerušení aplikace nedošlo ke ztrátě dat, či aby při ukončení aplikace byly správně ukončeny i jiné komponenty [27]. Životní cyklus *Activity* je ilustrován diagramem 3.7.

3.4.2 Services

V případě *Activity* byla komponenta přímo viditelná uživateli, kdežto komponenty *Services* běží na pozadí a jsou tak uživateli skryty. *Services* jsou aktivní po předem neurčený časový interval. Uživatel má možnost ovládat jejich běh pomocí interakce s *Activity*, ke které je cílená *Service* připojena. *Service* je možné připojit k libovolné aplikaci. Nemusí se tedy jednat bezprostředně o aplikaci, která *Service* spustila. Stejně tak jejich běh není podmíněný konkrétní aplikaci. Pokud aplikace, která *Service* spustila, není dále viditelná nebo byla ukončena, nemusí nutně *Service* skončit s ní, ale může pokračovat ve svém běhu dále. Implicitně *Service* běží ve stejném vlákně jako spouštěcí aplikace. Pokud je však vyžadován větší výpočetní výkon, který by běh aplikace zpomalil, je možné spustit *Service* v samostatném vlákně. Typickým příkladem využití *Services* je přehrávání hudebních souborů či stahování souborů [30].



Obrázek 3.7: Životní cyklus *Activity* [9]

3.4.3 Broadcast receivers

Broadcast recievers jsou komponentami, které přijímají všesměrové zprávy systému a reaguje na ně patřičnou akcí. Jedná se většinou o zprávy, jakými jsou stav baterie či displeje. Je na vývojáři aplikace, na které zprávy bude odpovídat a jakým způsobem. Typickým příkladem reakce na všesměrovou zprávu je zobrazení dialogového okénka, ve kterém je uživatel seznámen se vzniklou událostí a je mu umožněno na ni reagovat [23].

3.4.4 Komunikace komponent – Intents

Jednotlivé, výše uvedené komponenty spolu mohou a většinou také potřebují komunikovat. *Intents* se používají ke spouštění a přepínání kontextu *Activity* a *Services*. Jsou také využívány k zasílání zpráv *Broadcast recievers* a při komunikaci aplikací. Přenášené zprávy v sobě zapouzdřují informace o odesílateli, příjemci, volanou akci a její parametry [26]. K definici přenášených dat je mimo jiné možné využít následně popsanou komponentu, viz kap. 3.4.5.

3.4.5 Content Providers

Content Providers slouží jako univerzální prostředek k práci s daty aplikací a jejich sdílení. Dovolují ukládat, resp. číst data do, resp. z vnitřní paměti zařízení. K ukládání dat slouží vnitřní či externí souborový systém. Na ten jsou data ukládána v podobě souborů spravovaných samotným jádrem systému nebo jinými knihovnami, například relační databází SQLite. Poté, co jedna aplikace uloží svá data do systému, ostatní aplikace mohou pomocí různých typů *Content Providers* k datům této aplikace přistupovat a dále s nimi pracovat. *Content Providers* jsou jedinou možností sdílení dat mezi aplikacemi [25].

3.4.6 Manifest

Celá aplikace musí být doplněna popisem její struktury, vlastností a schopností. K tomu účelu je použit *Manifest*. Jedná se o strukturovaný XML soubor, který specifikuje, z jakých komponent se aplikace skládá (viz kap. 3.4), jakým zprávám rozumí (viz kap. 3.4.4), kterých knihoven využívá a v jakém balíčku[1] se nachází. Dále specifikuje práva pro přístup z jiných aplikací a vymezuje, které verze Androidu aplikace podporuje a zda je k jejímu běhu potřeba speciální hardware, např. GPS modul, či parametry mobilního zařízení (velikost displeje). Touto specifikací můžeme vymezit mobilní zařízení, na která je aplikace cílená a tak zajistit přenositelnost pouze na ta zařízení, která jsou podporována. Soubor dále definuje název a ikonu aplikace. Tento soubor je typicky pojmenován `AndroidManifest.xml` [32].

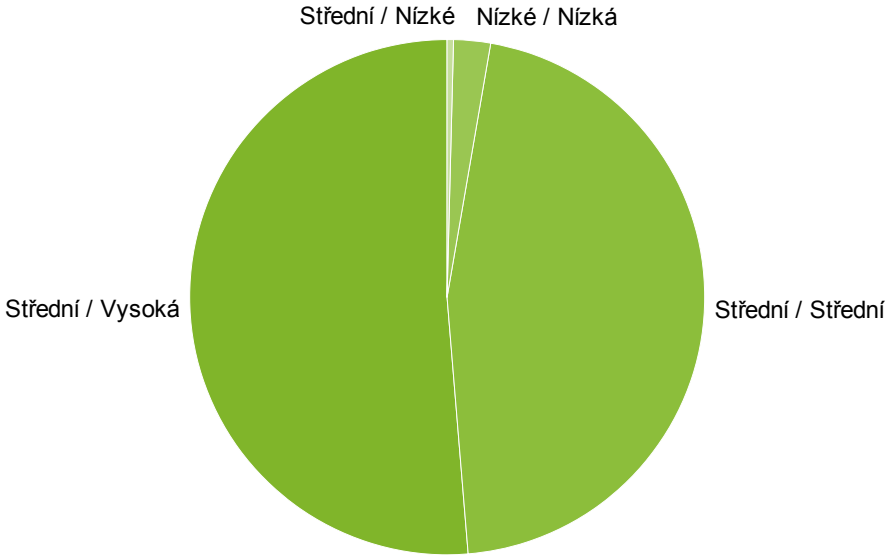
3.5 Odlišnosti verzí a hardwaru mobilních zařízení

Současný trh s mobilními zařízeními, která využívají platformu Android, je rozdělen mezi jednotlivé verze Androidu (viz obr. 3.3). Pokud má být aplikace úspěšná a hojně využívaná, nelze se z pohledu vývojáře aplikace zaměřit pouze na jednu konkrétní verzi, ale je potřeba postihnout co největší množství verzí a tedy co nejvíce mobilních zařízení. Z tohoto pohledu by vývojář měl používat programovací techniky a přístupy, které jsou dostupné ve většině verzí Androidu. Pokud je aplikace vytvářena k použití se speciálním hardwarem, například

[1] název balíčku slouží jako unikátní identifikátor aplikace

GPS modulem, je potřeba výslednou aplikaci omezit pomocí Manifestu (viz kap. 3.4.6) pouze na zařízení s potřebnou výbavou [24].

Dalším úskalím jsou i rozdílné vlastnosti displejů cílených mobilních zařízení. Displeje těchto zařízení jsou rozděleny do třech kategorií podle velikosti displeje. Každá kategorie se pak dělí do dalších tří kategorií dle hustoty obrazových bodů. Vývojář aplikace proto musí při implementaci s tímto rozdělením počítat a připravit grafický materiál (pozadí obrazovek, ikony tlačítek, atd.) pro všechny kategorie. K problému je možné přistupovat dvojí cestou. První z nich spočívá ve vytvoření a použití vlastního, vlastnoručně vytvořeného grafického materiálu. Druhou možností je jednotlivé prvky aplikace nechat vykreslit pomocí grafických tříd `Shapes` či `9-Patch`. S pomocí těchto tříd je možné vytvářet základní geometrické útvary, barevné přechody a stíny. Takto vytvořená grafika je pak automaticky přizpůsobena vlastnostem a možnostem konkrétního mobilního zařízení, resp. jeho displeje. Následující graf 3.8 reprezentuje rozdělení mobilních zařízení dostupných na trhu do kategorií dle parametrů jejich displejů. Údaje byly zaznamenány při přístupech zařízení do Android Marketu ve 14-ti denním intervalu od 20. července do 2. srpna 2010 [28, 31, 49].



Obrázek 3.8: Zastoupení rozlišení (první údaj) a hustot (druhý údaj) displejů mobilních zařízení na trhu – Nízké / Nízká 2,3 %, Střední / Nízká 0,4 %, Střední / Střední 45,9 %, Střední / Vysoká 51,2 % [12]

Kapitola 4

Návrh aplikace

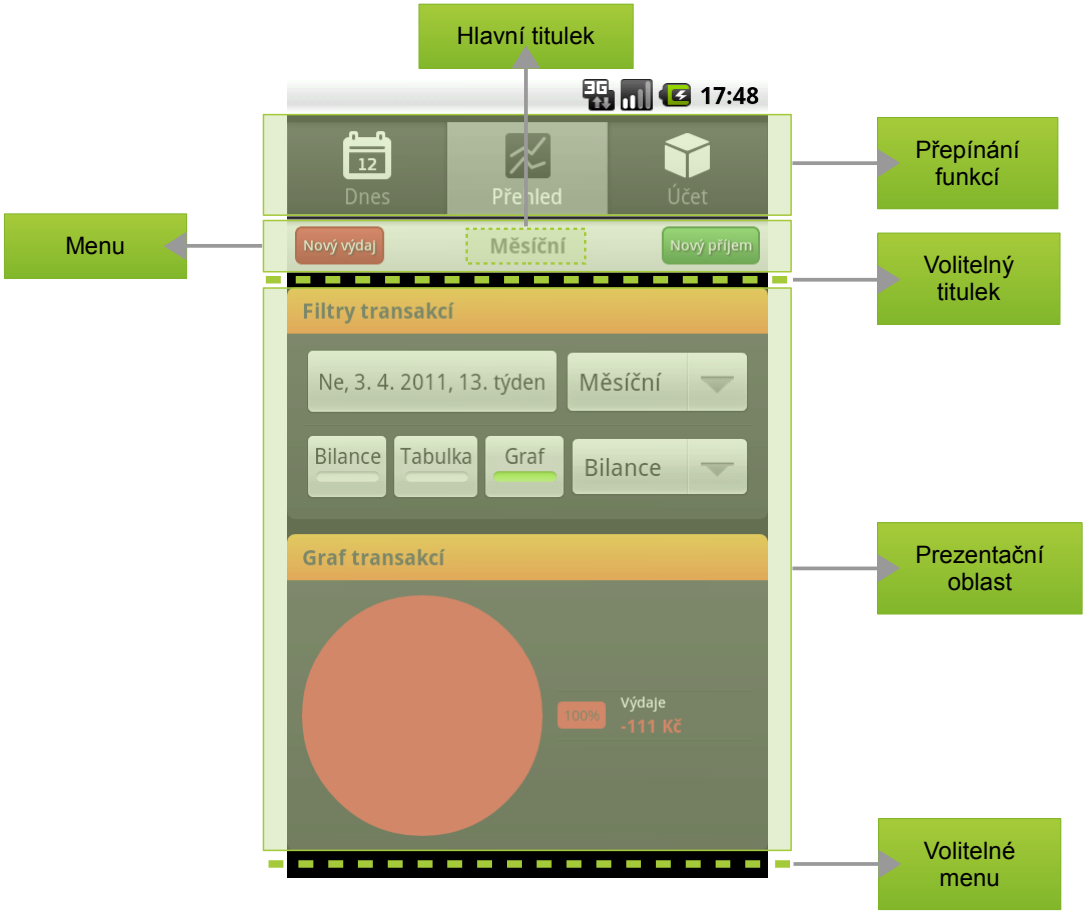
Kapitola návrh aplikace popisuje proces přípravy k samotné implementaci. Zabývá se rozvahami o možném vzhledu aplikace. Dále komentuje jednotlivé části uživatelského rozhraní a popisuje jejich funkčnost. V kapitole je také zmíněno, jakým způsobem se budou v aplikaci ukládat perzistentní data a na diagramech jsou ukázány možnosti jejich manipulace a vztahy mezi jednotlivými entitami.

4.1 Uživatelské rozhraní

Jedním z klíčových bodů zadání práce je požadavek na uživatelsky příjemné a intuitivní ovládání. Z tohoto požadavku návrh uživatelského rozhraní vychází. Aplikace je tak rozdělena do třech logických celků, ve kterých je zapouzdřena spolu související funkcionalita aplikace. Aplikace by měla být intuitivní a ovládací prvky aplikace by měly být zřetelné a jednoznačné. Ovládání aplikace je řešeno tak, aby uživatel nemusel využívat hardwarových tlačítek zařízení, ale aby měl všechny ovládací prvky přístupné přímo na displeji. Tento přístup má uživateli zpříjemnit a urychlit práci s aplikací. Doporučované přístupy, náměty a připomínky pro tvorbu aplikací a jeho uživatelského rozhraní nabízené přímo vývojáři Androidu jsou dostupné v [33, 5]. Samotný grafický návrh uživatelského rozhraní, rozložení ovládacích prvků a obsahu jednotlivých logických částí aplikace byl proveden formou nákresů a náčrtků na papír. Tato přibližná podoba byla následně převedena do elektronické podoby, na které byl následně proveden výběr barev prvků aplikace. Při výběru palety barev byla snaha zvolit takové barevné kombinace, aby byl celkový barevný ráz kontrastní a tedy i dobře čitelný. Příklad grafického vzhledu a rozmístění prvků aplikace je možné vidět na obrázku 4.1. Následující kapitoly blíže specifikují funkcionalitu jednotlivých částí (celků) aplikace.

4.2 Aktuální den

První logický celek v sobě zapouzdřuje funkcionalitu související s aktuálním kalendářním dnem. Tento celek uživateli slouží k rychlé správě transakcí, které náleží aktuálnímu kalendářnímu dni. Uživatel zde má možnost přidávat nově vzniklé transakce, upravovat již existující a zobrazovat podrobnosti o každé z nich. Jednotlivé transakce jsou zobrazeny v tabulce. Zde jsou klíčovými informacemi čas vzniku, kategorie, do které transakce spadá, způsob platby a hodnota transakce. Zobrazené transakce je možné řadit dle uvedených klíčových atributů (vyjma kategorií). V tabulce jsou zobrazovány i pravidelné a napláno-



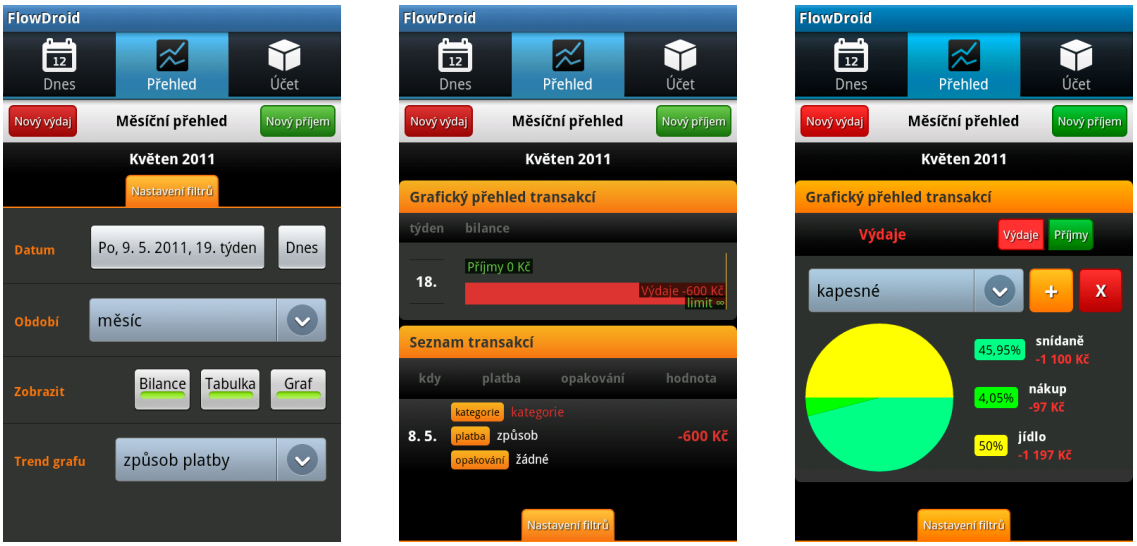
Obrázek 4.1: Ukázka možného vzhledu a rozložení prvků uživatelského rozhraní vytvářené aplikace

vané transakce s výskytem v aktuálním dni. Součástí celku je i bilance aktuálního dne, která má podobu tabulky obsahující součet příjmů, výdajů a jejích rozdílů. V bilanci probíhá i kontrola nastaveného limitů. Poslední funkcí této části je zobrazování uživatelem nastavených upozornění na výskyt transakce.

4.3 Přehledy transakcí

Druhým celkem jsou přehledy transakcí. Jak již název napovídá, celek uživateli poskytuje možnost zobrazovat přehledy transakcí za jím specifikované období. Základním specifikátorem je datum, ze kterého se má přehled generovat. Zvolené datum, například 1. 1. 2011, může zároveň označovat konkrétní den 1. leden nebo také 1. týden či měsíc v roce 2011. Uživatel tedy společně s referenčním datem vybírá i rozsah působnosti. Po specifikaci časového období si uživatel volí formát, v jakém chce údaje o transakcích zobrazit. Na výběr je zde několik možností. První dvě varianty přehledů jsou obdobou způsobu prezentace, jež je použita v předchozím celku (viz kap. 4.2). Novou možností je grafický přehled formou grafů. Při jeho volbě je možné specifikovat trend grafu. Nabídka sestává ze zobrazení bilance, výdajů a příjmů pro jednotlivé kategorie a platební metody za vybrané období.

Na základě vybraného trendu se automaticky volí typ použitého grafu. U bilance je použit sloupcový, u kategorií a platebních metod zase kruhový graf. Součástí přehledů transakcí je také možnost dodatečného přidání transakce ke zvolenému datu. Popsané funkce jsou ilustrovány obrázkem 4.2.



Obrázek 4.2: Ukázky funkcí logického celku přehledu transakcí

4.4 Uživatelské účty

Třetí a poslední částí je uživatelský účet. Hlavní funkcí uživatelského účtu je sdružování transakcí. Každá transakce náleží právě jednomu uživatelskému účtu. Tato asociace pak dovoluje transakcím globálně nastavit další parametry. Jedná se zejména o limity za určitá období, nebo o možnost nastavit transakcím jejich měnu. Zobrazení limitů, jejich aktuální hodnoty a stavu čerpání je další vlastností tohoto celku. U uživatelského účtu je také nabízena možnost přidání pravidelných transakcí, které se dle nastaveného intervalu budou opakovat. Speciální variantou této funkce je i možnost naplánování transakce v budoucnosti společně s nastavením upozornění na její výskyt. Tyto transakce jsou poté zobrazovány v tabulce, která poskytuje stejné funkce, jako ta v předchozích dvou částech. Aplikace podporuje více uživatelských účtů, a tak v tomto celku nalezneme i jejich správce, který nám umožní nové účty přidávat a upravovat. Ze seznamu účtů si pak budeme moci vybrat ten, který chceme právě používat. Správce účtu nabízí uživateli také možnost převodu měny účtu s možností automatické aktualizace převodního kurzu.

4.5 Vytváření transakcí

Všechny tři zmíněné logické celky v sobě obsahují funkci přidávání transakcí (příjmů i výdajů). Tuto funkci je možné chápat také jako zapouzdřený celek, neboť v sobě sdružuje jak možnost přidávání, tak i editaci transakcí. Použitá varianta se volí na základě požadavku uživatele. Přidávání, resp. úprava transakcí je koncipována jako formulář. Ke změnám jeho položek dochází na základě toho, který logický celek formulář vytvořil. Promítané změny

jsou hlavně v možnosti manipulace s nastavením data a voleb pro opakování a upozornění transakce, které jsou dostupné pouze při vytváření plánovaných transakcí, viz kap. 4.4.

4.6 Transakce

V předchozích kapitolách jsme několikrát zmiňovali pojem transakce. Transakce nám reprezentuje platbu a to v náš prospěch (příjem) a v neprospěch (výdaj). Pojmy transakce a platba lze tedy libovolně zaměňovat. Každá transakce sestává z identifikátoru a několika parametrů, které ji blíže specifikují. Jedná se zejména o datum a čas vzniku, hodnotu, ať už kladnou nebo zápornou, a způsob plnění transakce (platební metoda). Právě zmíněné atributy lze považovat za nejdůležitější a také povinné. Rozšiřujícími a volitelnými vlastnostmi transakce jsou místo vzniku transakce, popis transakce, příznak opakování a upozornění.

4.7 Kategorie

Podstatným rozšířením transakcí je možnost volby kategorií. Kategorie nabývají většího významu při zobrazování grafických přehledů, ve kterém si uživatel volí, jaké kategorie chce zobrazit. Může tak snadno odhalit, kam putuje většina finančních prostředků. Rozlišování kategorií je založeno jak na jejím názvu, tak barvě, která je implicitně zvolena dle bilance transakce - výdaj (zelená) nebo příjem (červená). Uživateli jsou kategorie nabízeny dle frekvence využití (používanější mají větší prioritu).

4.8 Uživatelský účet

Jak již bylo zmíněno v kapitole 4.4, účet slouží především k asociování s transakcemi a k definování jejich globálních vlastností. Těmito vlastnostmi jsou limity a měna transakcí. Aplikace poskytuje uživateli možnost tvorby více účtů a to z důvodu rozlišení skupin transakcí. Například pokud uživatel cestuje do zahraničí, bude chtít vědět pouze o transakcích provedených v cizině. Pohodlně tak může učinit výběrem jiného, k tomu určenému účtu.

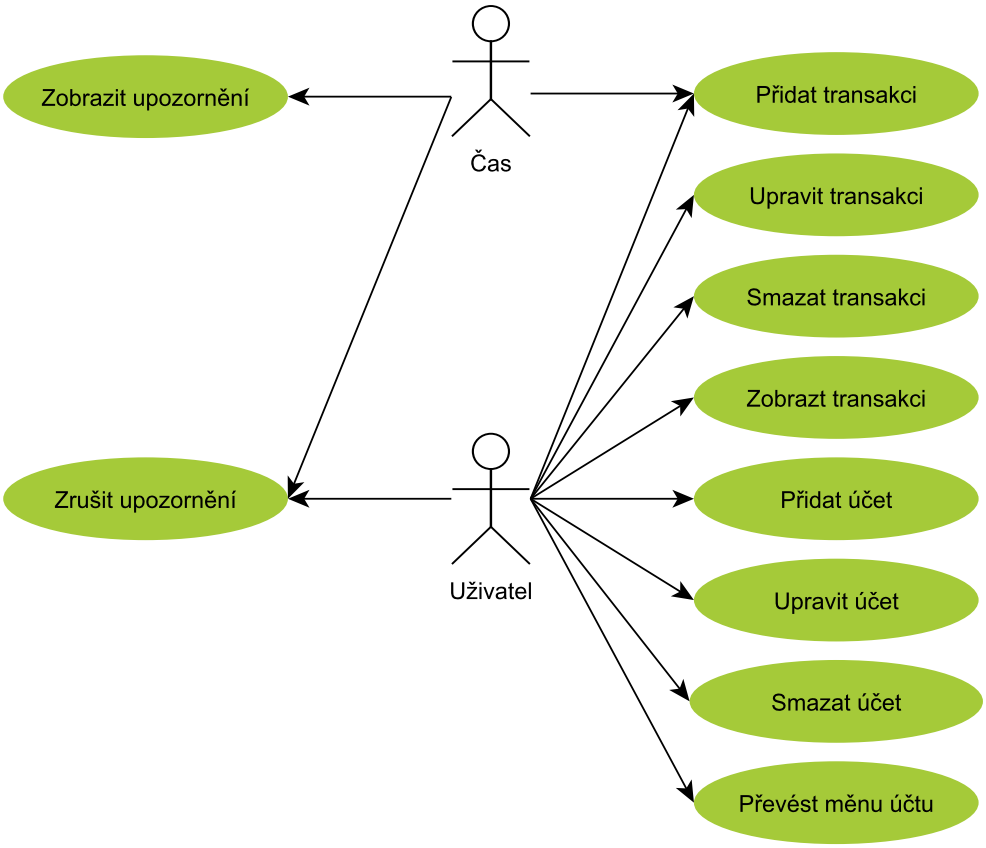
4.9 Ukládání a práce s daty

Možností, jak persistentně ukládat data v aplikaci existuje několik, viz kap. 3.4.5. K uložení aplikačních dat – transakcí, kategorií a uživatelských účtů se z nabízených možností jeví jako nejvýhodnější varianta relační databáze SQLite. Ta dovoluje mimo pohodlné a efektivní vkládání a čtení dat také možnost provádět nad daty agregační dotazy, které jsou vhodné zejména pro práci s transakcemi. Další pozitivní vlastností relační databáze je její schopnost udržovat data pomocí integritních omezení v konzistentním stavu. Možnosti použití aplikace a vztahy mezi jednotlivými entitami dat jsou ilustrovány následujícími diagramy.

4.9.1 Diagram případů užití aplikace

Nejčastější případy užití aplikace jsou vystiženy v diagramu 4.3. Některé případy jsou jistým zobecněním skutečné funkcionality aplikace. Přidání transakce v sobě zapouzdřuje jak přidávání běžných, tak i pravidelných (opakujících se) transakcí. Dalším zobecněním je zobrazení transakce, kterým je myšleno zobrazení detailu jedné transakce i přehledu skupin.

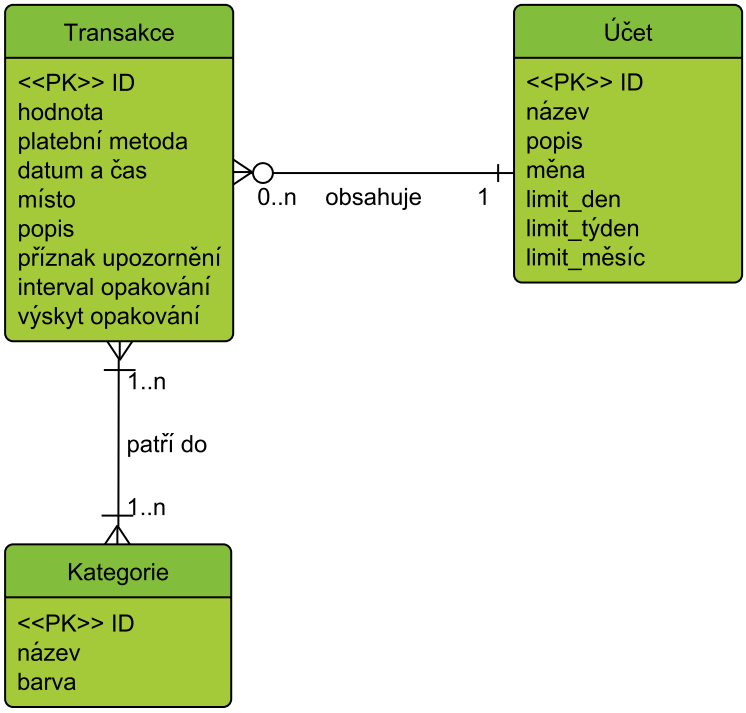
Hlavním aktérem je uživatel pracující s aplikací. Svou roli zde hraje také čas, který vyvolává časově závislé akce.



Obrázek 4.3: Diagram případů užití aplikace

4.9.2 Diagram vztahů mezi entitami dat

Vztah entit reprezentujících data uložená v databázi je znázorněn v diagramu 4.4. Účet a transakce jsou v relaci, která značí, že jednomu účtu může náležet několik transakcí, ale jedna transakce náleží vždy pouze jednomu účtu. Relace mezi transakcemi a kategoriemi naopak značí vztah, kdy jedna transakce může patřit do více kategorií a jedna kategorie může náležet více transakcím. Z jednotlivých entit můžeme také vyčíst jejich klíčové atributy.



Obrázek 4.4: Diagram vztahů mezi entitami databáze

Kapitola 5

Implementace

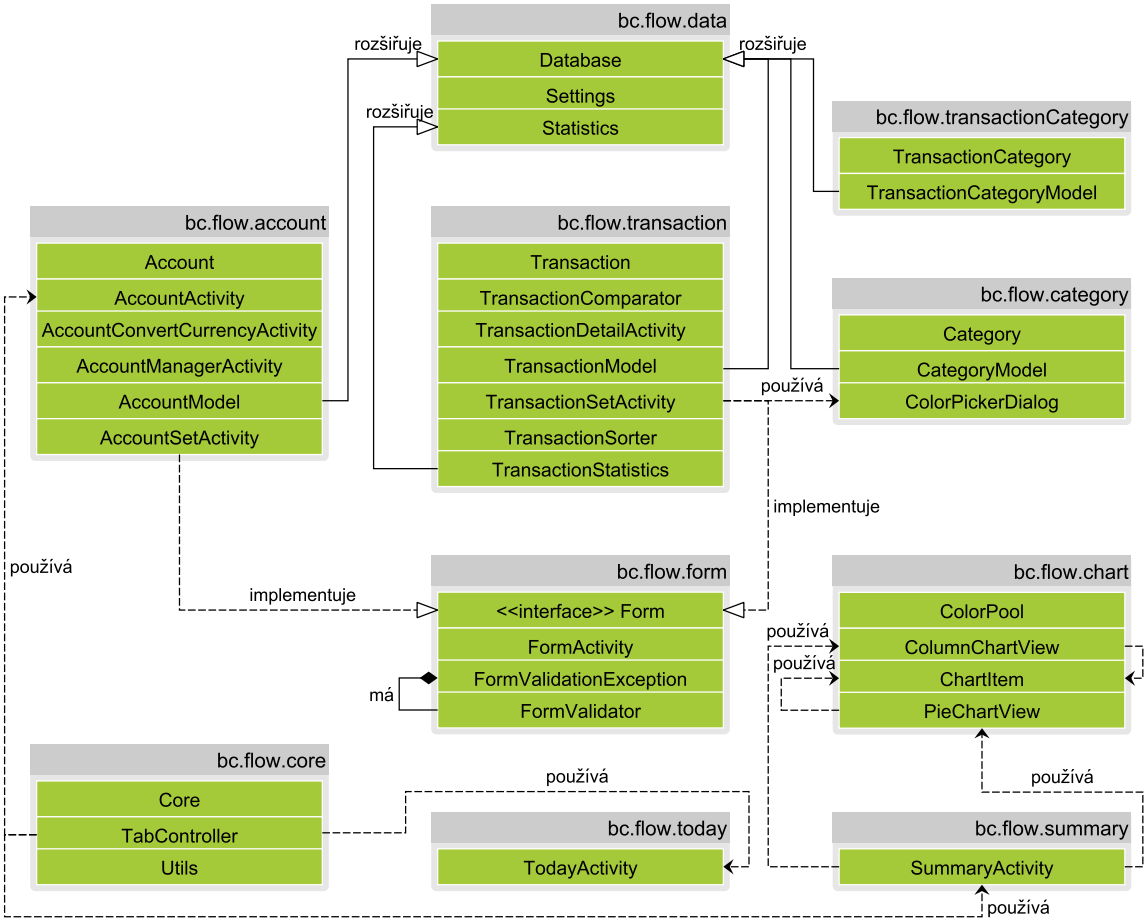
Tato kapitola se zabývá použitými postupy a metodami při tvorbě vyvíjené aplikace. Začátek je věnován programové struktuře a následně jsou blíže popsány její jednotlivé části. Obsah kapitoly je také zaměřen na objasnění řešení implementovaných funkcí aplikace a problémům popsaných v kapitole 4.

5.1 Struktura aplikace

Části aplikace jsou rozděleny do balíků, které v sobě sdružují spolu související funkcionalitu. Tento přístup výrazně napomáhá vývojáři při organizaci zdrojových kódů rozsáhlejších aplikací. K usnadnění vývoje jsou dále použity návrhové vzory, konkrétně MVC (*Model – View – Controller*), Singleton a Observer. Způsob jejich konkrétního užití je popsán v příslušných podkapitolách. Základem celé aplikace je balíček `bc.flow.core`, jemuž je věnována následující kapitola. Ostatní, závislé balíky jsou popsány v dalších částech kapitoly. Struktura aplikace je ilustrována diagramem na obrázku 5.1. Jsou v něm znázorněny příslušnosti jednotlivých tříd do konkrétních balíků a k dokreslení souvislostí sledovaných v následujících částech kapitoly jsou vyznačeny některé podstatné vztahy.

5.2 Jádro aplikace

Jádro aplikace je tvořeno dvěma hlavními a jednou vedlejší třídou. První z nich je třída `Core`. Jedná se o potomka třídy Androidu `Application` [4], která slouží k uchovávání globálních stavů aplikace. Tato třída a její potomci jsou automaticky inicializováni při spuštění aplikace. Konkrétně je volána metoda `onCreate()`, ve které je uveden veškerý efektivní kód. V našem případě je tato metoda použita jako úvodní inicializátor klíčových součástí aplikace. Z větší části se jedná o inicializaci tříd s návrhovým vzorem Singleton a to z důvodu jejich pohodlnějšího a rychlejšího využití v ostatních částech aplikace. Singleton je také použit z nutnosti poskytovat všem objektům stejný stav. Další využití třídy `Core` spočívá v načítání nastavení aplikace, připojení databáze a inicializace modelů pro její manipulaci. Obsahem `Core` jsou i proměnné, jež jsou využívány ve více částech aplikace, a to z důvodu optimalizace, kdy se předchází jejich opakované inicializaci a nastavení. Další hlavní součástí aplikačního jádra je třída `TabController` 5.4.1 sloužící k přepínání logických celků (viz kap. 5.4). Zmiňovanou vedlejší třídou je soubor globálně přístupných pomocných metod – `Utils`. Ta zahrnuje například funkce pro převod prvního znaku slova na jeho



Obrázek 5.1: Rozdělení částí aplikace do balíčků a nejvýznamnějších vztahů mezi třídami

velkou variantu či převod pole řetězců do textové podoby ve formátu hodnot oddělených a ohraničených zadanými znaky.

5.3 Aplikační data

K ukládání dat je využíváno relační databáze SQLite. Ta je úložištěm pro transakce, jejich kategorie a uživatelské účty. Základní operace (vytvoření a nastavení) s touto databází jsou zapouzdřeny v třídě `Database` 5.3.1. Manipulace s jejími tabulkami je obsahem databázových modelů jednotlivých ukládaných objektů. Trvale je ukládáno i nastavení aplikace, které má v režii třída `Settings` 5.3.2. Součástí balíčku `bc.flow.data` je společně se zmíněnými třídami také bázová třída pro tvorbu statistik `Statistics` 5.3.3.

5.3.1 Třída `Database`

Při prvním spuštění aplikace je vytvořena aplikační databáze, probíhá nastavení jejich parametrů a jsou vloženy její tabulky. Tato činnost je obsažena v třídě `Database`, která k části svého běhu využívá třídy `SQLiteOpenHelper`, jež je abstrakcí nad knihovnou relační databáze SQLite [4]. Třída `Database` je také využita jako bázová třída k vytvoření databázo-

vých modelů (část *Model* z návrhového vzoru MVC), které jsou dále schopny obsluhovat jednotlivé, jím náležející tabulky databáze. Při inicializaci modelu je touto třídou poskytnuta instance databáze, nad kterou jsou posléze modelem prováděny operace. Zároveň je modelům poskytnuta možnost řídit databázové transakce.

Transakce

Objekty třídy `Transaction` jsou komplexní reprezentací transakcí uložených v databázi. V aplikaci se využívají jak při vytváření a úpravách transakcí, tak k jejich zobrazování. Pro tyto účely třída obsahuje mimo hodnoty obsažené v transakci také několik výčtových konstant přidávajících možnosti voleb pro vlastnosti transakcí, kterými jsou perioda opakování (denně, týdně a měsíčně), bilance (příjem a výdaj) a proveditelné operace (přidání, editace, zobrazení detailu a smazání). `Transaction` obsahuje také metody pro generování výskytů pravidelných transakcí v používaných obdobích (den, týden a měsíc). Při generování výskytů nedochází k žádnému vkládání nových záznamů do databáze, ale pouze ke klonování původní (výchozí) transakce s novým datem a časem. Tyto metody jsou používány při zobrazování přehledů.

Ke komunikaci s databází, konkrétně přílušnou tabulkou, je používán model `TransactionModel`. Ten obsahuje metody pro vkládání, rušení a editaci záznamů. U těchto metod je odchytáván koncový stav operace a je vracen jako výsledek volající straně, která patřičně upozorní uživatele. Součástí modelu jsou také metody pro načítání transakcí dle různých parametrů, kterými mohou být identifikátor transakce, uživatelského účtu nebo časový údaj. Model dále obsahuje metodu `getMethods()` používanou pro získání agregace platebních metod ve formuláři pro vytváření a editaci transakcí, viz kap. 5.4.3. Návratovým typem těchto metod jsou objekty třídy `Transaction` nebo přímo databázový kurzor.

Kategorie

Abstrakcí nad kategoriemi transakcí uloženými v databázi je třída `Category`. Objekty této třídy jsou používány při práci s transakcemi, neboť jsou jejich nedílnou součástí. Dále se využívají při filtrování dat v grafických přehledech. Práci s tabulkou databáze zastřešuje model `CategoryModel`.

Uživatelský účet

Uživatelský účet je reprezentován třídou `Account`. Ta je hlavním datovým objektem logického celku uživatelského účtu – `AccountActivity`. Definice stejně jako u předchozích dvou tříd zahrnuje konstanty pro názvy sloupců databázové tabulky a řetězec pro vytvoření tabulky samotné. Tyto konstanty jsou využívány při interakci s databázovým modelem, kterým je zde `AccountModel`.

5.3.2 Třída `Settings`

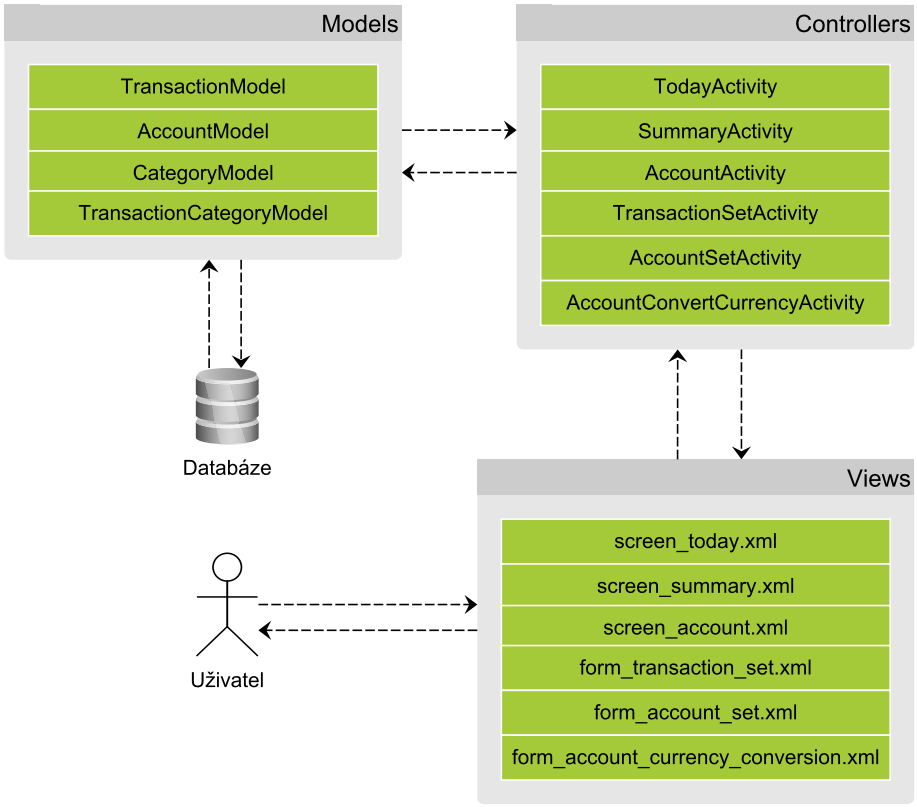
Tato třída slouží k načítání a ukládání nastavení aplikace. K tomuto účelu využívá typ datového úložiště *Shared Preferences*. Ukládány jsou hodnoty primitivních datových typů ve formě páru klíč – hodnota, jenž je zapsán do souboru uloženého v trvalé (stabilní) paměti mobilního zařízení [4]. Aplikací ukládaná data jsou identifikátor aktivního uživatelského účtu a jeho měna. Jejich hodnoty jsou následně použity k nastavení kontextu, resp. stavu, ve kterém uživatel s aplikací pracuje.

5.3.3 Třída `Statistics`

Při tvorbě bilance a grafických přehledů je využíváno statistiky transakcí, ze které se generují zobrazované hodnoty. Třída `Statistics` takovou statistiku vytváří. Hlavními údaji jsou součty příjmů, výdajů, jejich suma, počet započtených transakcí a extrémy těchto hodnot. Třída rovněž slouží jako základ pro pokročilejší statistiky transakcí obsažených ve třídě `TransactionStatistics`.

5.4 Implementace logických celků

Celá aplikace je navržena (viz kap. 4) a rozdělena do tří logických celků (částí). Jednotlivé části jsou tvořeny třídou `Activity` [4], resp. jejím rozšířením. Pro přehled se jedná o třídy `TodayActivity` , `SummaryActivity` a `AccountActivity`. Celky jsou výchozími body k obsluze celé aplikace. Jedná se o prvky *Controller*, které na základě definic vzhledu (*View*) zobrazují uživateli data načtená databázovými modely (*Model*). Na základě interakce uživatele s aplikací pak zpracovávají vstupy a opět pomocí modelů ukládají případné změny. Tímto způsobem je v aplikaci využit návrhový vzor MVC. Základní funkcionalita celků je rozšiřována dalšími samostatnými častmi. Jedná se především o práci s transakcemi 5.4.3 a uživatelskými účty 5.4.5. Schéma na obrázku 5.2 ilustruje popsaný způsob využití návrhového vzoru MVC.



Obrázek 5.2: Schéma využití návrhového vzoru MVC v aplikaci

5.4.1 Přepínání logických celků

Logické celky je možné mezi sebou přepínat pomocí záložek. K tomuto účelu je v aplikaci použita třída `TabController`, která je rozšířením třídy Androidu `TabActivity` [4]. `TabActivity` slouží k přepínání vložených komponent, jakými mohou být *Activity* nebo pouze definice jejich vzhledů *View*. Současně je při přepínání záložek kontrolováno nastavení aplikace, konkrétně existence vybraného uživatelského účtu. Pokud není žádný účet vybrán, je uživatel odkazován do správce účtů (viz kap. 5.4.5), dokud tak neučiní. Přesněji dochází nejprve k přepnutí kontextu na záložku účet a z ní je teprve spuštěn správce. Jak již název napovídá, `TabController` je zároveň zástupce objektů aplikace, které reprezentují *Controller* z návrhového vzoru MVC (viz kap. 5.4).

5.4.2 Uživatelské rozhraní

Uživatelské rozhraní logických celků je tvořeno souborem prezentačních a ovládacích prvků systematicky rozmístěných na obrazovce aplikace (viz obr. 4.1). K definici požadovaného rozložení je použit hierarchický popis XML souborem. Soubor obsahuje kořenový element určující způsob rozložení, jakým se v něm zanořené prvky mají řadit. Těmito prvky jsou opět další skupiny rozložení či již samostatné koncové elementy, jako jsou například textová pole, bitmapy či části formuláře (vstupní pole, rozbalovací menu, tlačítka). Definice používaných předpisů částí uživatelského rozhraní jsou umístěny ve složce `res/layout`.

Vzhled prvků a jejich pozice vůči ostatním je nastavována pomocí atributů. Společné vlastnosti skupin objektů jsou nadefinovány ve stylovacím souboru `style.xml`, který plní stejnou funkci jako kaskádové styly v jazyce HTML. Ty dovolují z jednoho místa ovládat vlastnosti více elementů příslušejících do cílené skupiny. Stylování v Androidu také podporuje dědičnost, jež je využívána k vytváření závislostí ve skupinách. Podřazené prvky rozšiřují vlastnosti nadřazených a nemusí znovu definovat již zděděné atributy. Odchylky od zavedených stylů je možné zavést přímo do definice objektu v předpisu vzhledu uživatelského rozhraní a to díky vyšší prioritě takto uvedeného záznamu před záznamem ze souboru stylů. K nastavení globálního vzhledu aplikace a logických celků je využito speciální varianty stylů – témat. Konkrétně jsou používána témata `Theme.Black` a `Theme.Black.NoTitleBar`. Ze samotného názvu plyne, že se jedná o témata vzhledu tmavého rázu ve variantách s, resp. bez titulku, který zobrazuje, resp. nezobrazuje název aplikace. Varianta bez názvu je určena ke stylování logických celků vložených do přepínače záložek 5.4.1, který název aplikace již obsahuje.

Problém různorodosti parametrů displejů mobilních zařízení zmíněný v kapitole 3.5 je řešen několika doporučenými technikami uvedenými v [31]. V aplikaci, resp. jeho uživatelském rozhraní, není v žádném ohledu (definice elementů, jejich vzhled a rozmístění) použito absolutních hodnot a rozměrů. Na místo jsou použity relativní jednotky a hodnoty míry, na základě kterých aplikace automaticky výslednou velikost a rozměr dopočítává. Dalším použitým přístupem k řešení problematiky variability displejů je tvorba objektů uživatelského rozhraní pomocí grafických tříd `Drawable` a `ShapeDrawable` [4]. Ty tvoří objekty na základě kreslící plochy (canvas), která je specifická pro konkrétní displej a jeho parametry. Pomocí zmíněných tříd lze objekty vykreslovat buď z definice v XML souboru, nebo čistě programově. Definiční přístup je použit ke generování barevných přechodů ve statických objektech (hlavičky a pozadí oblastí dat). Tímto přístupem jsou generována také tlačítka v menu a formulářích (přidávání a odstraňování kategorií u transakcí a grafu). Tyto definice jsou uloženy v adresáři `res/drawable`. Programový přístup je využíván k tvorbě dynamických součástí grafických přehledů. Zde je při výpočtu rozměrů a pozic grafických objektů

zahrnut koeficient hustoty `density` displeje dostupný v třídě `DisplayMetrics` [4]. Jeho použití nám zajišťuje požadovanou nezávislost na parametrech displeje. Grafický materiál, který není možné pro svou složitost generovat pomocí právě uvedených postupů, je vlastnoručně vytvořen pomocí externího (není součástí SDK) grafického editoru. Jedná se o ikony aplikace a záložek, které jsou uloženy ve složkách `res/drawable-*`, kde `*` zastupuje verzi API (viz kap. 3.3) a kategorii displeje (viz kap. 3.5).

Součástí uživatelského rozhraní jsou mimo výše uvedené prvky také různé typy konstant. Všechny textové popisky, ať už název aplikace, záložek, nadpisů, tlačítek a zpráv jsou centralizovány v podobě řetězcových konstant v náležitých XML souborech ve složce `res/values`. Konstanty jsou sdružovány buď v seznamech, anebo polích. Obdobně jsou realizovány hodnoty používaných barev, animací, měn a položek kontextových menu. K parsování konstant z XML souborů do programových proměnných slouží třída `Resources` [4]. Její instance je v aplikaci globálně dostupná v rámci třídy `Core` (viz kap. 5.2).

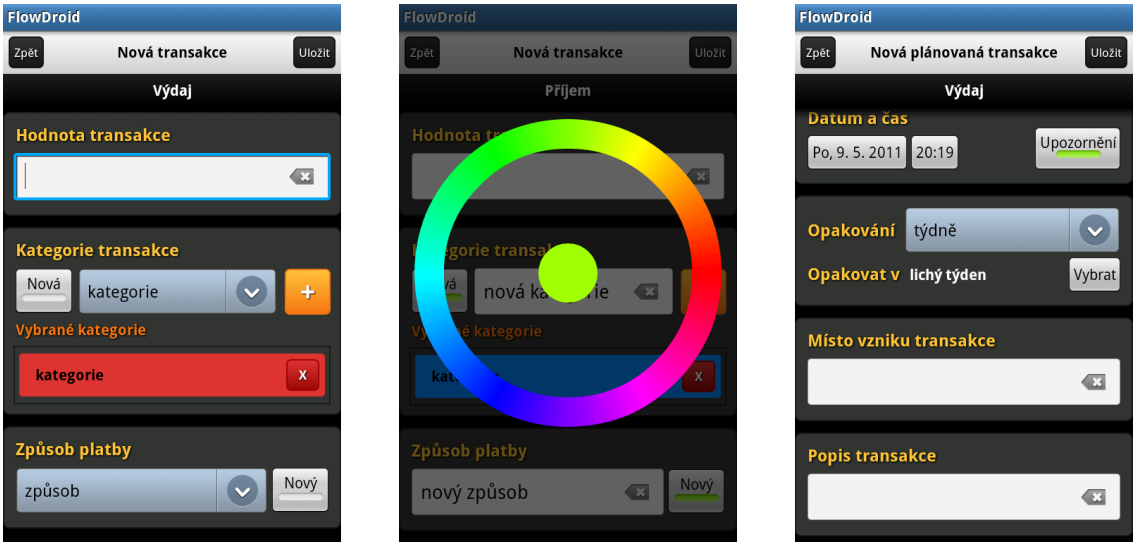
5.4.3 Vytvoření a editace transakcí

Vytváření a editace transakcí je, stejně jako jiné části aplikace určené k interakci s uživatelem, reprezentována komponentou *Activity*. Jedná se tak o další *Controller*. Jeho implementace vychází z abstraktní třídy `FormActivity`, která poskytuje podporu pro tvorbu formulářů. Ta zahrnuje prostředky k ověření správného vyplnění vstupních polí (obstarává třída `FormValidator`) a rozšiřuje možnosti editace vloženého textu – pole jsou doplněna o možnost rychlého smazání obsahu stiskem dodatečné ikony v pravé části. O obsluhu funkce ikony se stará třída `clearInputOnTouchListener`, jehož definice je součástí `FormActivity`. Její funkčnost je založena na principu detekce stisknuté oblasti vůči poloze ikony. Součástí formulářové *Activity* je také předepsané rozhraní `Form`, které mimo jiné definuje metodu `validate()` starající se o zmíněné ověření vyplnění formuláře. V případě chybného vyplnění je uživatel upozorněn příslušnou zprávou o chybě v místě jejího výskytu. Zmiňovaná rozšíření a s nimi související třídy jsou obsahem balíčku `bc.flow.form`.

Tvorba a úprava transakce je zapouzdřena ve třídě `TransactionSetActivity`. Konkrétní vyvolaná akce (vytvoření nebo úprava transakce) je vybrána na základě požadavku uživatele a varianta jejího formuláře je zvolena v závislosti na kontextu, ve kterém požadavek vzniká. Možnými případy jsou přidávání a editace transakcí pro aktuální kalendářní den či datum vybrané v přehledech transakcí. Tento případ vyústí ve variantu formuláře bez možnosti manipulace s datem vzniku a volbou pro upozornění a opakování transakce. Důsledkem tohoto omezení je možnost přidávat pouze aktuální nebo staré transakce. K plánování transakcí je určen logický celek *Účet*, který také povoluje nastavení upozornění a opakování.

Součástí formuláře transakce je i možnost správy kategorií, které vlastního správce nemají. Přidání nové kategorie je realizováno v rámci jejího zadání a přidání do vybraných kategorií transakce. Volbu barvy je možné provést jak při vytváření, tak při úpravě transakce, a to stisknutím pole reprezentujícího příslušnou kategorii. Uživateli se poté zobrazí dialog pro výběr barvy. Stejné pole obsahuje také tlačítko k odstranění kategorie. Změna názvu kategorie možná není, neboť dopad na ostatní transakce, konkrétně jejich význam a identifikaci uživatelem by mohl být neočekávaný. Pokud chce uživatel transakci přiřadit jinou, neexistující kategorii, musí vytvořit novou. Mazání kategorií je prováděno automaticky (metoda `deleteZombies()` v modelu `CategoryModel`) a to v případě, že kategorie již není nadále přiřazena žádné transakci. Současně je ošetřena integrita databáze a to pomocí integritních omezení u cizích klíčů patřičných tabulek. Popisované části formuláře jsou

ilustrovány obrázkem 5.3.



Obrázek 5.3: Části formuláře pro přidání a úpravu transakce

5.4.4 Zobrazení transakce a přehledů

V aplikaci je použito několik variant (forem) prezentace vložených transakcí. Nejzákladnější z nich je zobrazení detailu transakce – obstarává třída `TransactionDetailActivity`. Uživatel má v případě potřeby možnost z jejího prostředí transakci upravit. `TransactionDetailActivity` zachovává kontext, ve kterém byla *Activity* spuštěna a tak jej může při volbě editace distribuovat dále do formuláře a vybrat tím jeho odpovídající formu (viz kap. 5.4.3). Detail transakce se vyvolává v kontextovém menu, které je dostupné pro jednotlivé řádky tabulky transakcí. Realizace tohoto a dalších, komplexnějších forem prezentace, jsou popsány v následujících odstavcích.

Bilance

Bilance slouží jako rychlý přehled objemu uskutečněných transakcí. Zobrazuje ve formě tabulky celkové příjmy a výdaje, stav čerpání limitu a konečnou bilanci v podobě sumy příjmů a výdajů. Bilance je využita v logických celcích `TodayActivity` a `SummaryActivity` (viz kap. 5.4). Definice jejího vzhledu se nachází v souboru `widget_balance.xml`. Tento soubor slouží jako znovupoužitelná komponenta, která je před každým vykreslení inicializována, jsou nastaveny hodnoty jejich elementů, a poté je přidána do kořenového uzlu, který je umístěn v definici vzhledu pro celý logický celek.

Údaje pro tvorbu bilancí jsou obsaženy ve statistikách. Zaznamenané hodnoty jsou při vykreslování testovány proti pozitivnímu významu pro uživatele, a pokud nabývají nepříznivých hodnot (překročení limitu a záporná bilance) jsou patřičně, typicky červeně označeny. Aktualizace hodnot je prováděna ve stejném vlákně jako aplikace, neboť se nejedná o výpočetně náročný proces a jeho průběh nesnižuje odezvu uživatelského rozhraní.

Graf

Grafické přehledy transakcí jsou implementovány jako potomci základní komponenty uživatelského rozhraní – třídy `View` [4], která poskytuje prostředky pro vykreslování objektů a obsluhu jejich událostí. Komponenty jsou kompletně tvořeny programátorským přístupem a jako datové položky používají objekty třídy `GraphItem`, která je jistou formou abstrakce nad komplexní třídou `Transaction`, viz kap. 5.3.1. Objekty `GraphItem` obsahují název reprezentovaných dat, hodnotu v podobě statistik a barvu, jež má být použita k vykreslení. Aplikace obsahuje dvě varianty grafických přehledů.

První variantou je sloupcový graf používaný k zobrazování agregací hodnot transakcí za vybrané období (bilance). Agregace hodnot je tvořena třídou pokročilých statistik transakcí `TransactionStatistics` (viz kap. 5.3.3). Implementace samotného grafu se nachází v třídě `ColumnchartView`. Jádro třídy je tvořeno metodou `onDraw()`. Ta má na starost vykreslení obsahu komponenty, která zahrnuje sloupce hodnot (příjmy a výdaje), jejich popis a aktuální hranici zbývajícího limitu. Rozložení hodnot transakcí mezi kategorie a platební metody je vizualizováno pomocí kruhového grafu, který je druhou variantou grafického přehledu transakcí. Vykreslování je řešeno v třídě `PieChartView`. Celý kruhový graf vznikne postupným skládáním jednotlivých výsečí reprezentujících odpovídající část dat. Uživatel má při prezentaci transakcí kruhovým grafem možnost zvolit, zda ho zajímají příjmy nebo výdaje (implicitní volba). Při vizualizaci kategorií si rovněž ze seznamu vybírá požadované kategorie. U platebních metod jsou zobrazeny všechny dostupné metody (zde je předpoklad omezeného počtu metod).

Obě varianty grafových komponent jsou pouze části celkového zobrazení grafických přehledu. To obsahuje i legendu grafu, která je tvořena jinou komponentou, konkrétně XML definicí, z níž se části legendy tvoří a následně přidávají k celku. Výsledek je vložen do boxu přehledu opatřeného nadpisem a hlavičkou.

Jelikož by tvorba přehledu s větším počtem dat mohla značně výpočetně zatížit aplikaci a uživatel by tak mohl pociťovat sníženou odezvu, je celá část vykreslování spouštěna v novém vlákně a průběh procesu je po dílčích částech zobrazován uživateli. K tomuto účelu byla použita třída `AsyncTask` [4], která pomocí svých metod dovoluje jak manipulaci s uživatelským rozhraním (manipulace možná pouze z hlavního vlákna aplikace), tak spuštění a správu nového vlákna pro výpočty. Vykreslování popsaných komponent je obsaženo v `SummaryActivity` a jejich vnitřních třídách `InvalidateColumnChart` a `InvalidatePieChart`.

Tabulka

Zobrazení v tabulce poskytuje uživateli chronologický přehled transakcí. Každá položka s sebou nese časový údaj o vzniku, asociované kategorie, způsob úhrady, interval opakování a hodnotu transakce. U jednotlivých záznamů (řádků) je možné vyvolat kontextové menu, které nabízí zobrazení detailu, úpravu a smazání vybrané transakce. Vybrání volby provede příslušnou akci ať už v rámci aktuální *Activity* nebo v jiné, nově spuštěné. Záznamy je možné řadit dle zmíněných údajů transakce (mimo kategorii). Mechanismus řazení je zapouzdřen v třídě `TransactionSorter`, která k porovnávání hodnot využívá nadefinované komparátory třídy `TransactionComparator`. Porovnávání je založeno na návrhovém vzoru Observer. Jeho princip spočívá v reakci na stisk pole v hlavičce tabulky, které dle svého identifikátoru nastaví patřičný komparátor, směr řazení (mění se s každým kliknutím) a o změně informuje zaregistrované posluchače (*Observers*). Jejich registrace a samotné přidání řazení do tabulky je prováděno v konstruktoru třídy řazení. Posluchač pak ve své

metodě `update(...)` provede patřičnou akci, jež je zpravidla aktualizace tabulky s transakcemi.

Tabulka transakcí je díky své schopnosti manipulace s transakcemi použita ve všech třech logických celcích, ve kterých zobrazuje specifická data (transakce pro aktuální den – `TodayActivity`, dle vybraného data – `SummaryActivity` či plánované – `AccountActivity`). Vzhled tabulky je definován v souboru `table_transaction.xml`. Jednotlivé záznamy jsou pak tvořeny z komponent definovaných v souboru `table_transaction_entry.xml`. K jejich vykreslení je, stejně jako v případě grafických přehledů, z důvodu optimalizace výkonu a odezvy aplikace, použito asynchronního zpracování v odděleném vlákně (viz kap. 5.4.4).

5.4.5 Správce účtů

Součástí logického celku uživatelského účtu je správce účtů, který poskytuje uživateli operace, jakými jsou vytváření, editace, mazání, výběr (přepínání) a převod měny účtu. Poslední tři zmíněné akce jsou dostupné v rámci kontextového menu, jež je k dispozici pro každý účet zobrazený v seznamu. Přidání nového účtu je spouštěno samostatným tlačítkem.

Nový účet a úprava existujících

Vytváření a editace uživatelských účtů je řešeno stejně jako v případě transakcí pomocí `FormActivity`. Její výhody jsou komentovány v kapitole 5.4.3. Formulář obsahuje vstupní pole pro název (unikátní a povinná položka), měnu, limity a popis účtu. Aktuální verze aplikace podporuje měny zahrnuté v kurzovním lístku České národní banky [2]. Změna měny účtu je možná pouze pomocí funkce převodu, viz následující kapitola.

Převod měny

Uživatelské účty jsou pro komplexnost doplněny o možnost převodu jejich měny. Formulář převodu obsahuje dvě vybírané položky. Těmi jsou cílová měna a převodní kurz. Ten je možné zadat ručně nebo jej aktualizovat ze serveru. K načítání převodního kurzu je použito rozšíření *Kalkulačka* ze stránky iGoogle, která poskytuje API pro převod měn. To je voláno přes metodu GET protokolu HTTP. Následně je z odpovědi, v podobě JSON řetězce, extrahován a nastaven převodní kurz.

5.4.6 Tabulka limitů

Celek `AccountAcitivty` obsahuje mimo tabulkový přehled plánovaných transakcí (popsaný v 5.4.4) také stav nastavených limitů účtu. Ten je zobrazován v tabulce. Její jednotlivé řádky zahrnují popis a stav čerpání limitu pro používaná období (den, týden a měsíc). V případě překročení zvoleného limitu je o této skutečnosti uživatel informován zvýrazněním příslušného údaje. Vzhled tabulky je definován v souboru `table_limits.xml`. Její hlavička pak v `table_limits_header.xml` a řádky v `table_limits_row.xml`.

5.5 Testování

Aplikace byla otestována na třech odlišných fyzických zařízeních. Ty se lišily jak hardwarovým vybavením, tak verzí Androidu. Konkrétně se jednalo o Sony Ericsson Xperia

X10 mini[1], který disponuje velmi malým rozlišením a hustotou displeje a byl tak ideálním kandidátem k testování vzhledu a použitelnosti uživatelského rozhraní. Rovněž byla v jeho prostředí testována odezva aplikace. Dalším testovacím prostředkem byl HTC Wildfire[2], na kterém byla provedena stejná sada testů jako u předchozího zařízení. Dalšímu testování byla aplikace podrobena na Samsung Galaxy S[3]. Ten reprezentuje špičku mezi mobilními telefony, a tak sledovanými faktory zde byla, namísto vzhledu a výkonnosti aplikace, její stabilita a obecně výskyt chyb. Aplikace byla v průběhu vývoje spouštěna ve virtuálním zařízení (emulátoru), a tak během uvedených cyklů testování na reálných zařízení byly odhaleny na dvě desítky chyb, které se během vývoje aplikace v emulátoru neprojevily. Nejčastější chyby se týkaly uživatelského rozhraní, jež nabývalo, díky úpravám výrobců konkrétních zařízení, rozdílného vzhledu, než v neupravené verzi dostupné v emulátoru. Častým chybám podléhala také samotná struktura a logika kódu, kdy docházelo při specifické posloupnosti provedených akcí k pádu aplikace z důvodu špatné inicializace a alokace zdrojů.

Testována byla i intuitivnost a celková práce s uživatelským rozhraním. Testovanými subjekty byly vlastníci zmíněných mobilních zařízení. Ti byli po seznámení se schopnostmi aplikace požádáni o provedení sérií úkolů. Tyto úkoly zahrnovaly základní úkony s aplikací, konkrétně uživatelskými účty (vytvoření, výběr) a transakcemi (přidání, editace, mazání a zobrazení přehledů). Po dokončení testů byl se subjekty proveden dialog, jehož cílem bylo zhodnocení provedených testů. Vyhodnocení poskytlo cenné informace o názoru na vytvořenou aplikaci. Velmi dobře byla hodnocena pohodlnost a rychlost práce s aplikací, jež je považována za nejdůležitější vlastnost jejího uživatelského rozhraní. Subjektům se také líbil vzhled aplikace a uvítali prostředí v českém jazyce.

[1] Sony Ericsson Xperia X10 mini: verze Android 2.1, 600 MHz procesor, 176 MB RAM, rozlišení displeje 240 x 320 obr. bodů, velikost displeje 2,55 palců [6]

[2] HTC Wildfire: verze Android 2.2, 528 MHz procesor, 384 MB RAM, rozlišení displeje 240 x 320 obr. bodů, velikost displeje 3,2 palců [37]

[3] Samsung Galaxy S: verze Android 2.2, 1 GHz procesor, 512 MB RAM, rozlišení displeje 480 x 800 obr. bodů, velikost displeje 3,2 palců [51]

Kapitola 6

Závěr

Poslední kapitola zprávy k bakalářské práci na téma Osobní plánovač financí pro operační systém Android shrnuje průběh práce a snaží se porovnat dosažené výsledky s požadovanými. Část kapitoly se zaobírá i budoucností aplikace, možnostmi jejího rozšíření v návaznosti na její současný stav a jiné bakalářské projekty v tomto roce.

6.1 Přístup ke splnění zadání

K vypracování bakalářské práce dle bodů uvedených v zadání bylo potřeba nejprve provést analýzu aktuálně dostupných a používaných prostředků ke správě osobních financí. S přihlédnutím k povaze a cílům práce byly analyzovány aplikace pro přední platformy mobilních zařízení. Snažení v tomto ohledu je shrnuto v kapitole 2. Následně bylo přistoupeno ke studiu platformy Android, zahrnujícím jak seznámení s její konkurencí, tak platformou samotnou. Mimo historii a obecnou koncepci Androidu jsem se také seznámil se způsoby a možnostmi vývoje aplikací. Průběh a výsledek studia je obsahem kapitoly 3. Návrh aplikace, jež zahrnoval rozčlenění aplikace do celků, rozmyšlení jejich funkcí a nákres uživatelského rozhraní je popsán v kapitole 4. Této části byl věnován velký důraz, aby se tak předešlo možným problémům při implementaci. Struktura aplikace společně s jejími důležitými a podstatnými aspekty implementace je obsažena v 5. V závěru této kapitoly je také zmíněn průběh testování. Naplněním těchto kroků bylo splněno zadání v plném rozsahu.

6.2 Výsledná aplikace

Výsledkem práce je aplikace, jež má jejím uživatelům sloužit jako nástroj ke správě jejich financí, poskytnout jim možnost finance kontrolovat a plánovat. Předpokladem úspěšného plánování osobních financí je nutnost zaznamenat každou proběhlou transakci. Ke splnění tohoto cíle je aplikace navržena tak, aby se dobře a pohodlně používala a uživatel měl s přidáváním transakcí co nejméně práce. Důraz byl kladen i na samotný vzhled aplikace a uživatelského rozhraní, jež by mělo být intuitivní a moderní. Z tohoto pohledu jsou prvky uživatelského rozhraní rozmístěny a stylovány tak, aby co nejvýstižněji reprezentovaly jejich význam. Za účelem moderního vzhledu jsou použity barevné přechody, stíny a animace. Aplikace se tak snaží držet krok s aktuálními trendy uživatelských rozhraní. Mezi hlavní funkce aplikace patří přidávání příjmů a výdajů, systém kategorií transakcí, tabulkové a grafické přehledy, nastavení a hlídání limitů, vytváření pravidelných transakcí

s možností upozornění na výskyt a podporu více uživatelských účtů s nastavitelnou měnou a jejím převodem.

Cílem vývoje bylo také docílit co nejširšího využití aplikace a tak s ohledem na aktuální podíl verzí Androidu na trhu mobilních telefonů (viz obr. 3.3) jsou podporovány majoritní verze 2.1, 2.2 a s výhledem do budoucna také 2.3. Aplikace podporuje všechny skupiny používaných rozlišení těchto verzí, viz kap. 3.5. Mezi další pozitivní vlastnosti aplikace je možné považovat její velikost[1] a velmi dobrou odezvu i na slabších zařízeních. Tato tvrzení byla otestována na třech fyzických zařízeních a jsou tedy podložena reálnými výsledky. V porovnání s existujícími komerčními aplikacemi (viz kap. 2.2), určených k osobnímu plánování financí pro platformu Android, tak vytvořená aplikace nabízí srovnatelnou funkcionalitu, výkonnost a konkurenceschopný vzhled. Oproti analyzované konkurenci navíc aplikace poskytuje, pro některé uživatele zásadní věc, český jazyk.

6.3 Budoucnost a rozšíření aplikace

Do budoucna by bylo vhodné aplikace doplnit o internetový portál, který by uživateli poskytoval podporu osobního plánování i na PC. Přenos aplikace do prostředí stolních počítačů nabízí další možnosti širšího využití v podobě kooperace s jinými aplikacemi.

Vhodným příkladem se jeví bakalářská práce Martina Lutonského [44] na téma Osobní plánovač, který by mimo svoji funkčnost mohl sloužit také jako prostředek k zobrazování aplikací naplánovaných transakcí.

Dalším zajímavým rozšířením aplikace by byla možnost zaznamenávat GPS souřadnice pro každou transakci a to buď přímo z GPS modulu obsaženému v telefonu nebo z informací fotografie z místa vzniku transakce. Z těchto údajů by pak bylo možné generovat mapu transakcí. Jednotlivé záznamy by tak dosahovaly větší informační hodnoty, jež je možné využít například při tvorbě cestovních výkazů. Případným rozšířením by také mohla být možnost načtení informací o transakcích na základě detekce textu ve vybraných oblastech fotografie účtenky, faktury či jiného platebního dokladu.

[1] v závislosti na verzi Androidu dosahuje aplikace velikosti v řádech několika stovek kilobytů – přibližně 400 kB

Literatura

[1] Apple: Apple [online]. http://www.apple.com, 2011 [cit. 2010-04-23].

[2] Česká národní banka: Česká národní banka *Kurzy devizového trhu* [online]. http://www.cnb.cz/cs/financni_trhy/devizovy_trh/kurzy_devizoveho_trhu/denni_kurz.jsp, 2011 [cit. 2011-05-01].

[3] BlackBerry: BlackBerry [online]. http://us.blackberry.com/ataglance, 2011 [cit. 2010-04-23].

[4] Developers, A.: Android Developers *Reference – Class Index* [online]. http://developer.android.com/reference/classes.html, 2011 [cit. 2011-04-25].

[5] Developers, A.: Android *User Interface Design Tips* [online]. http://www.slideshare.net/AndroidDev/android-ui-design-tips, 2011 [cit. 2011-04-25].

[6] Ericsson, S.: Sony Ericsson Xperia x10 mini *Specifications* [online]. http://www.sonyericsson.com/cws/corporate/products/phoneportfolio/specification/xperiax10mini, 2011 [cit. 2010-05-01].

[7] Foundation, T. A. S.: Apache Harmony. http://harmony.apache.org, 2011 [cit. 2011-03-17].

[8] Gartner: Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012 [online]. http://www.gartner.com/it/page.jsp?id=1622614, 2011 [cit. 2010-04-23].

[9] Google: Android Developers *Activity lifecycle* [online]. http://developer.android.com/guide/topics/fundamentals.html#actlife, 2010-12-22 [cit. 2010-12-28].

[10] Google: Android Developers *Android 2.2 Platform Highlights* [online]. http://developer.android.com/sdk/android-2.2-highlights.html, 2011 [cit. 2011-03-18].

[11] Google: Android Developers *Android 3.0 Platform Highlights* [online]. http://developer.android.com/sdk/android-3.0-highlights.html, 2011 [cit. 2011-03-18].

[12] Google: Android Developers *Screen Sizes and Densities* [online]. http://developer.android.com/resources/dashboard/screens.html, 2011 [cit. 2011-03-18].

[13] Google: Android Developers *Android 1.5 Platform* [online]. http://developer.android.com/sdk/android-1.5.html, 2011 [cit. 2011-04-24].

[14] Google: Android Developers *Android 1.6 Platform* [online]. http://developer.android.com/sdk/android-1.6.html, 2011 [cit. 2011-04-24].

[15] Google: Android Developers *Android 2.1 Platform* [online]. http://developer.android.com/sdk/android-2.1.html, 2011 [cit. 2011-04-24].

[16] Google: Android Developers *Android 2.2 Platform* [online]. http://developer.android.com/sdk/android-2.2.htmll, 2011 [cit. 2011-04-24].

[17] Google: Android Developers *Android 3.0 Platform* [online]. http://developer.android.com/sdk/android-3.0.html, 2011 [cit. 2011-04-24].

[18] Google: Android Developers *What is Android?* [online]. http://developer.android.com/guide/basics/what-is-android.html, 2011 [cit. 2011-04-24].

[19] Google: Nexus S *Pure Google* [online]. http://www.google.com/nexus, 2011 [cit. 2011-04-24].

[20] Google: Open Handset Alliance *Members* [online]. http://www.openhandsetalliance.com/oha_members.html, 2011 [cit. 2011-04-24].

[21] Google: Open Handset Alliance [online]. http://www.openhandsetalliance.com/, 2011 [cit. 2011-04-24].

[22] Google: Android Developers *Activities* [online]. http://developer.android.com/guide/topics/fundamentals/activities.html, 2011 [cit. 2011-04-25].

[23] Google: Android Developers *Application Fundamentals* [online]. http://developer.android.com/guide/topics/fundamentals.html, 2011 [cit. 2011-04-25].

[24] Google: Android Developers *Backward Compatibility for Applications* [online]. http://developer.android.com/resources/articles/backward-compatibility.html, 2011 [cit. 2011-04-25].

[25] Google: Android Developers *Content Providers* [online]. http://developer.android.com/guide/topics/providers/content-providers.html, 2011 [cit. 2011-04-25].

[26] Google: Android Developers *Intents and Intent Filters* [online]. http://developer.android.com/guide/topics/intents/intents-filters.html, 2011 [cit. 2011-04-25].

[27] Google: Android Developers *Managing the Activity Lifecycle* [online]. http://developer.android.com/guide/topics/fundamentals/activities.html#Lifecycle, 2011 [cit. 2011-04-25].

[28] Google: Android Developers *MultiResolution - Multiple Resolutions* [online]. http://developer.android.com/resources/samples/MultiResolution/index.html, 2011 [cit. 2011-04-25].

[29] Google: Android Developers *SDK* [online]. http://developer.android.com/sdk/index.html, 2011 [cit. 2011-04-25].

[30] Google: Android Developers *Services* [online]. http://developer.android.com/guide/topics/fundamentals/services.html, 2011 [cit. 2011-04-25].

[31] Google: Android Developers *Supporting Multiple Screens* [online]. http://developer.android.com/guide/practices/screens_support.html, 2011 [cit. 2011-04-25].

[32] Google: Android Developers *The AndroidManifest.xml File* [online]. http://developer.android.com/guide/topics/manifest/manifest-intro.html, 2011 [cit. 2011-04-25].

[33] Google: Android Developers *User Interface Guidelines* [online]. http://http://developer.android.com/guide/practices/ui_guidelines/index.html, 2011 [cit. 2011-04-25].

[34] Google: Android Developers *User Interface* [online]. http://developer.android.com/guide/topics/ui/index.html, 2011 [cit. 2011-04-25].

[35] Google: Android Developers *Platfrom Versions* [online]. http://developer.android.com/resources/dashboard/platform-versions.html, 2011 [cit. 2011-05-05].

[36] Google: Android Developers *Android 2.3 Platform* [online]. http://developer.android.com/sdk/android-2.3.html, 2011 [cit. 2011-05-1].

[37] HTC: HTC Wildfire *Specification* [online]. http://www.htc.com/uk/product/wildfire/specification.html, 2011 [cit. 2010-05-01].

[38] HTC: HTC *Quietly Brillian* [online]. http://www.htc.com/www/htcsense/index.html, 2011 [cit. 2011-04-24].

[39] iBear: iBearMoney [online]. http://ibearmoney.com, 2011 [cit. 2011-04-23].

[40] iBear: Money 5 [online]. http://itunes.apple.com/us/app/money-5/id408818815?mt=8, 2011 [cit. 2011-04-23].

[41] IDC: Android Rises, Symbian∧3 and Windows Phone 7 Launch as Worldwide Smartphone Shipments Increase 87.2 % Year Over Year, According to IDC [online]. http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22689111&sectionId=null&elementId=null&pageType=SYNOPSIS, 2011 [cit. 2010-04-23].

[42] IDC: IDC Forecasts Worldwide Smartphone Market to Grow by Nearly 50 % in 2011 [online]. http://www.idc.com/getdoc.jsp?containerId=prUS22762811, 2011 [cit. 2010-04-23].

[43] Klega, V.: Nejlepší SMARTPHONY. *Chip*, ročník 21, č. 3, 2011: s. 50–53, ISSN 1210-0684, burda Praha, spol. s.r.o.

[44] Lutonský, M.: *Osobní plánovač*. Diplomová práce, Fakulta infromačních technologií Vysokého učení technického v Brně, 2010.

[45] Mobileware, F.: iXpenseIt *Take control of your finance today!* [online]. http://www.fyimobileware.com/ixpenseit.php, 2011 [cit. 2011-04-23].

[46] Mobileware, F.: iXpenseIt (Expense + Income = Cashflow with Budget) [online]. http://itunes.apple.com/app/ixpenseit-expense-income-cashflow/id284947174?mt=8, 2011 [cit. 2011-04-23].

[47] Pageonce: The Future of Money & Bills is Here [online]. http://www.pageonce.com, 2011 [cit. 2011-04-23].

[48] Pageonce: Pageonce Pro - Money & Bills [online]. https://market.android.com/details?id=com.netgate.android&feature=top-paid, 2011 [cit. 2011-04-23].

[49] Petřek, P.: Android: Styles *What´s not New, but is Undiscovered for many* [online]. http://www.slideshare.net/pavelpetrek/google-developer-day-2010-prague-styles-in-android, 2010-11-16 [cit. 2011-04-25].

[50] Pohjolainen, J.: Mobile Platform Overview [online]. http://www.slideshare.net/pohjus/mobile-platforms-in-general, 2010-10-22 [cit. 2010-04-23].

[51] Samsung: Samsung Galaxy S *Specifikace* [online]. http://www.samsung.com/cz/consumer/mobile-phone/mobile-phone/touchphone/GT-I9000HKAXEZ/index.idx?pagetype=prd_detail&tab=specification, 2011 [cit. 2010-05-01].

[52] Sayed Hashimi, D. M., Satya Komatineni: *Pro Android 2*. Apress, 2010, ISBN 978-1-4302-2659-8.

[53] StatCounter: StatCounter *GlobalStats* [online]. http://gs.statcounter.com/, 2011 [cit. 2011-04-23].

[54] Ziegler, C.: Engadget *Nokia taking over Symbian development, Foundation responsible for licensing* [online]. http://www.engadget.com/2010/11/08/nokia-taking-over-symbian-development-foundation-responsible-fo/, 2010-11-08 [cit. 2010-04-23].

Seznam příloh

Příloha A Obsah DVD

Příloha A

Obsah DVD

Přiložené DVD obsahuje následující materiály.

Technická zpráva
: soubor `Zpráva.pdf`

Zdrojové kódy technické zprávy
: adresář `./sources-tex`

Zdrojové kódy aplikace
: adresář `./sources-app`

Uživatelská nápověda
: soubor `Nápověda.pdf`

Pokyny a soubory ke zprovoznění aplikace
: soubor `README.txt` a adresář `./prerequisites`